Universidad Carlos III de Madrid
Escuela Politécnica Superior
Máster en Ciencia y Tecnología Informática
Especialidad en Inteligencia Artificial
Planning and Learning Group

Integración de Planificación de Alto Nivel, Ejecución y Percepción en Robots Autónomos

Trabajo de Fin de Máster

por

José Carlos González Dorado

8 de septiembre de 2014

Tutor:
Fernando Fernández Rebollo

TRABAJO DE FIN DE MÁSTER

INTEGRACIÓN DE PLANIFICACIÓN DE ALTO NIVEL, EJECUCIÓN Y PERCEPCIÓN EN ROBOTS AUTÓNOMOS

Autor: **JOSÉ CARLOS GONZÁLEZ DORADO**
Tutor: **FERNANDO FERNÁNDEZ REBOLLO**

TRIBUNAL

Presidente: **FUENSANTA MEDINA DOMÍNGUEZ**
Secretario: **SERGIO PASTRANA PORTILLO**
Vocal: **JOSÉ ANTONIO IGLESIAS MARTÍNEZ**

Tras el acto de defensa y lectura el día .... de Septiembre de 2014 en la **Escuela Politécnica Superior** de la **Universidad Carlos III de Madrid** (Leganés), el tribunal le otorga la siguiente **CALIFICACIÓN**:

Agradecimientos

Es en estos momentos en los que algo bonito termina, cuando echo la vista atrás y recuerdo la cantidad de veces que podría haber tomado otros caminos. Siempre ha habido alguien que, de algún modo, me ha ido llevando hasta donde estoy ahora. Si esas personas hubieran faltado, probablemente este trabajo nunca se habría escrito. Agradezco al PLG en general y a Fernando en particular la confianza que pusieron en mí desde el principio. Aún me cuesta creer la cantidad de cosas que he aprendido de todos ellos durante este año. También agradezco el apoyo de mis padres, de Sandra y de mis amigos, que aunque lo nieguen, siento que mi deuda con ellos es infinita. Y muy especialmente quiero agradecer a mi tocayo y amigo José Carlos todos estos grandes años de aventura y risas. Francamente, que también hayamos llegado juntos hasta aquí me parece surrealista. Ya hay ganas de ver qué nuevas locuras depara el futuro. Con gente como vosotros solo puedo imaginar lo mejor.

Email

josgonza@inf.uc3m.es

Teléfono

+34 91 624 5981

Dirección

Universidad Carlos III de Madrid

Escuela Politécnica Superior

Avda. de la Universidad, 30 - Lab. 2.1.B16

28911 Leganés (Madrid) - SPAIN

Por favor, cita este trabajo como:

José Carlos González: *Integración de Planificación de Alto Nivel, Ejecución y Percepción en Robots Autónomos*, Master thesis, Universidad Carlos III de Madrid, 2014.

Abstract

This work presents the design and implementation of part of an ad-hoc architecture for a humanoid, social and autonomous NAO robot. It is part of a project called NAOTherapist and its objective is to supervise upper-limb rehabilitation sessions for kids with Obstetric Brachial Plexus Palsy and Cerebral Palsy. The developed prototype plans the exercises to train in the therapy. After that, the robot manages a rehabilitation session performing the exercises itself. The patient has to imitate it to train the therapeutic objectives. Social and autonomy characteristics of this robot have required a heavy amount of modelling, implementation and original research [Pulido et al. 2014]. The design and implementation to achieve a functional prototype have derived in three parts for this work: the therapy designer with classical Automated Planning, the robot control component and a novel facial expression classifier.

Keywords

Social Assistive Robotics, SAR, rehabilitation, Classical Automated Planning, cognitive architecture, facial expressions, relational Automated Learning.

Resumen

En este trabajo se ha diseñado e implementado parte de una arquitectura ad-hoc para controlar un robot NAO humanoide, social y autónomo. Se enmarca dentro de un proyecto denominado NAOTherapist y su objetivo es para supervisar sesiones de rehabilitación de las extremidades superiores para niños con Parálisis Braquial Obstétrica y Parálisis Cerebral. El prototipo desarrollado planifica primero qué ejercicios se harán durante la terapia. Después el robot lleva el control de una sesión de rehabilitación realizando él mismo los ejercicios para que el paciente le imite. Las características sociales y de autonomía de este robot han requerido una fuerte carga de modelado e implementación, así como de investigación original [Pulido et al. 2014]. El diseño y la implementación para alcanzar un prototipo funcional han derivado en tres partes claramente diferenciadas: el diseñador de terapias con Planificación Automática clásica, el componente de control del robot terapeuta y un reconocedor de expresiones faciales.

Palabras clave

Robótica social de asistencia, SAR, rehabilitación, Planificación Automática clásica, arquitectura cognitiva, expresiones faciales, Aprendizaje Automático relacional.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se introduce el contexto que enmarca al proyecto NAOTherapist, que es el resultado de este trabajo. También se explican los protocolos terapéuticos seguidos y se motiva cada una de las partes desarrolladas. Finalmente, se recogen de forma concisa los objetivos concretos de este trabajo.

Dada la envergadura del proyecto, el equipo de desarrollo de NAOTherapist está constituido por José Carlos Pulido y el autor de este trabajo. El autor es responsable de una parte de la arquitectura, que es la que se presenta en esta memoria. No obstante, las numerosas dependencias entre los componentes de la arquitectura han obligado a que los dos miembros hayan tenido como mínimo un grado de implicación del 20 % en algunos de los componentes, aunque no hayan sido directamente de su responsabilidad.

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Economía y Competitividad (MINECO), con fondos de los proyectos coordinados THERAPIST[1] (TIN2012-38079-C03-01, TIN2012-38079-C03-02 y TIN2012-38079-C03-03), y FEDER INNTERCONECTA ADAPTA 2012[2]. El protocolo terapéutico que se sigue está basado en el del Hospital Universitario Virgen del Rocío (Sevilla).

[1] http://www.therapist.uma.es/ (Accedida el 2/9/2014)
[2] http://www.innterconectaadapta.es/ (Accedida el 2/9/2014)

1.1. Contexto

La Parálisis Braquial Obstétrica (PBO) es una pérdida de movimiento o debilidad de las extremidades superiores producida cuando la colección de nervios alrededor del hombro se daña durante el nacimiento (Figura 1.1). Este grupo de nervios se llama plexo braquial. Los avances en las técnicas del parto han reducido drásticamente la incidencia de la PBO, de modo que solamente 1,5 de cada 1.000 nacimientos presentan esta lesión y requieren rehabilitación. La parálisis braquial también puede producirse por accidentes o infecciones. En todos estos casos se recomienda terapia física de rehabilitación para recuperar total o parcialmente el rango de movimientos de los brazos afectados [Limthongthang et al. 2013].

Por otro lado, la Parálisis Cerebral (PC) es una enfermedad cerebral que puede impedir tener un control completo de las funciones motoras [Krägeloh-Mann et al. 2009]. El 60 % de los niños con deficiencias motoras está diagnosticado con esta enfermedad. Puede estar causada por complicaciones durante el parto y también por infección o accidente. De cada 1.000 nacimientos hay dos o tres casos que presentan esta sintomatología y se estima que 650.000 familias europeas tienen un niño o un adulto con parálisis cerebral [Castelli 2011].

Las dos enfermedades están directamente relacionadas con las extremidades superiores e inferiores, limitando la autonomía del paciente en tareas comunes como coger objetos, vestirse, comer, etc. No existe curación total para ellas, por lo que los pacientes tienen que aprender a vivir toda su vida con esta discapacidad. Sin embargo estas habilidades se pueden mejorar drásticamente mediante terapia de rehabilitación [Hensey 2009]. El cerebro de los niños es mucho más maleable y la rehabilitación, además de fortalecer los músculos, sirve para que el cerebro aprenda cómo mover mejor las extremidades afectadas. Por ello es recomendable empezarla desde temprana edad.

Este trabajo se centra en este conjunto de pacientes: niños de tres a catorce años

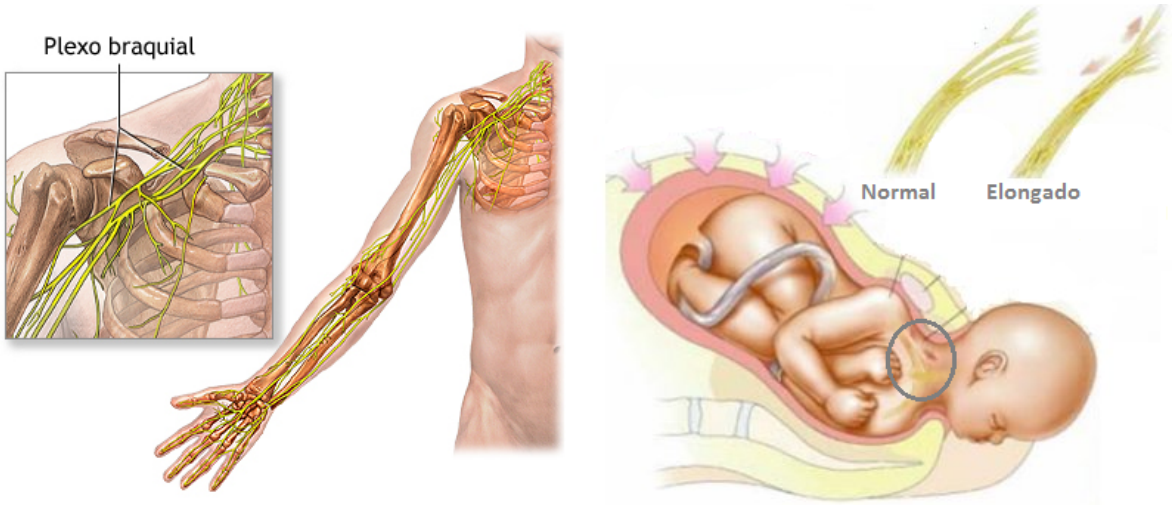

Figura 1.1: Plexo braquial y su elongación por movimientos bruscos durante el parto.

que puedan seguir una terapia de rehabilitación con un terapeuta sin contacto físico. El médico es el que determina qué pacientes son aptos para este tipo de terapia, excluyendo a aquellos que tengan problemas cognitivos graves, visión limitada o un rango de movimiento demasiado pequeño.

Desgraciadamente, las terapias son largas y a menudo aburridas, consistentes en continuos y repetitivos movimientos de brazos para entrenar las extremidades afectadas. Esta rutina de ejercicios tiende a causar la pérdida de interés en algunos pacientes. En particular, los niños requieren mucha más atención por parte de los terapeutas para evitar que se distraigan, lo que se traduce en un menor aprovechamiento de la rehabilitación [Calderita et al. 2013].

Además, a pesar de estos esfuerzos no siempre se alcanzan los objetivos de la terapia. La pérdida de motivación y compromiso para continuar el tratamiento puede obstruir la recuperación de forma crítica y afectar a la calidad de vida del paciente y de su familia. Esto también es negativo desde el punto de vista económico y profesional, debido al alto coste del personal especializado [Meyer-Heim et al. 2013]. Por ello, las técnicas que estimulen la atención y la motivación de estos pacientes durante su rehabilitación son muy necesarias.

1.2. Terapia de rehabilitación

Para este trabajo se ha tenido acceso directo al protocolo terapéutico interno del Hospital Universitario Virgen del Rocío, en Sevilla. No existe ninguna directriz universal sobre cómo deben llevarse a cabo estas terapias, pero el método de este hospital es fácilmente generalizable ya que estos protocolos varían poco y las métricas del progreso del paciente se basan en medidas estandarizadas.

En esencia, una terapia de rehabilitación se compone de unas 20 sesiones de 20 minutos aproximadamente, cada una compuesta por diferentes ejercicios. Cada ejercicio está compuesto, a su vez, de diferentes movimientos con los brazos para que el paciente los haga sentado o de pie. Existen tres actores principales involucrados en el protocolo terapéutico: el paciente, el médico y el terapeuta.

La Figura 1.2 muestra un diagrama de este protocolo terapéutico. El médico diagnostica al paciente, determina el número de sesiones de la terapia de rehabilitación y algunas restricciones para evitar el entrenamiento de ciertos grupos de ejercicios. También decide qué objetivos terapéuticos deben ser entrenados durante una terapia de rehabilitación. Existen cinco objetivos diferentes:

- Coordinación bimanual: coordinar mejor los movimientos que requieren el uso de ambas manos.
- Actividades unimanuales finas: mayor precisión de los movimientos que requieran una sola mano.
- Actividades unimanuales gruesas: fortalecimiento de todo el brazo.
- Posicionamiento del brazo: conseguir mayor amplitud de movimientos con el brazo.
- Posicionamiento de la mano: conseguir mayor precisión colocando la mano en el espacio.

El paciente tendrá que asistir a las sesiones de rehabilitación para aumentar su movilidad, fuerza y flexibilidad en las extremidades superiores. Las sesiones son secuen-

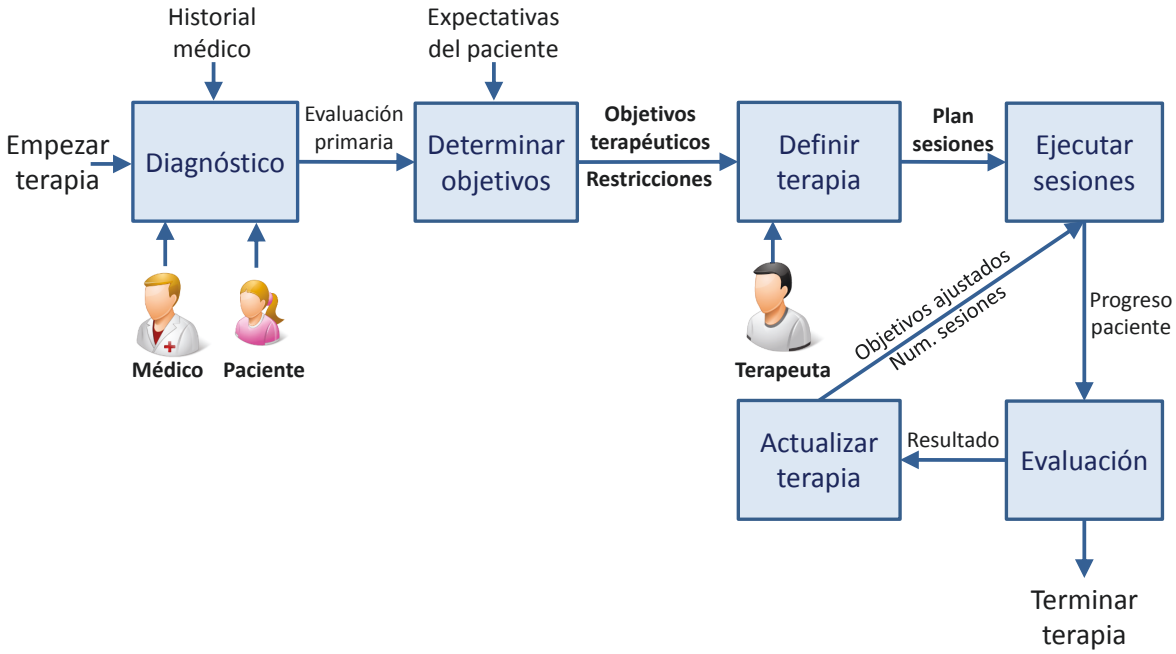

Figura 1.2: Esquema del protocolo terapéutico del Hospital Universitario Virgen del Rocío.

ciales y tienen una duración máxima y mínima.

El objetivo principal del terapeuta es ayudar al paciente a realizar las sesiones de rehabilitación mientras evalúa la evolución de la terapia. Cuando se planifica una sesión de rehabilitación, el terapeuta selecciona los ejercicios que, de acuerdo a su experiencia, son mejores para cumplir con los objetivos terapéuticos en un tiempo determinado. El Hospital Virgen del Rocío sigue unas guías generales internas que categorizan ejercicios de ejemplo de acuerdo a ciertas capacidades motrices que entrenan. Sin embargo, el terapeuta es libre de usar su creatividad para improvisar otros ejercicios que según él cumplan mejor las metas que propone el médico, intentando mantener la variedad para evitar que las sesiones sean demasiado monótonas.

Una sesión razonable puede estar organizada como sigue: los ejercicios iniciales sirven como calentamiento, los más intensos son realizados en la parte central de la sesión y los finales sirven de enfriamiento. Idealmente, la curva de intensidad de los

ejercicios a lo largo de una sesión se parecería a una función gaussiana.

Tras un cierto número de sesiones, se realizan evaluaciones periódicas para comprobar la evolución del tratamiento, tal como muestra la Figura 1.2. El progreso del paciente a lo largo de la terapia se mide usando la escala estandarizada GAS (*Goal Attainment Scale*) [Turner-Stokes 2009].

1.3. Motivación

NAOTherapist es un sistema de diseño y supervisión de terapia de rehabilitación en el que, tras planificar los ejercicios que se van a entrenar, un robot autónomo y humanoide va guiando al paciente, mostrándole los movimientos que debe realizar y corrigiéndole en caso de que sea necesario. Por un lado, el robot aumenta el interés y el compromiso del paciente con la terapia y por otro, ahorra tiempo de personal especializado. La idea es que sirva como herramienta de apoyo a la terapia, pero que no sustituya completamente a un terapeuta humano.

Este trabajo servirá como prototipo y prueba de concepto para el proyecto THERAPIST [Calderita et al. 2013], aún en las primeras fases de desarrollo. Utilizará el *framework* robótico RoboComp debido a que está previsto que también se utilice en THERAPIST. Para el prototipo NAOTherapist, el robot NAO[3] es un buen sustituto comercial del robot grande y con forma de oso de THERAPIST. El proyecto THERAPIST desarrollará mucho más la interacción social y se utilizará en un hospital real. Esto requiere una importante carga de investigación de la cual NAOTherapist formará parte.

El trabajo que se presenta en este documento puede dividirse en tres grandes secciones: el diseñador de terapias, el componente de control del robot terapeuta y el reconocedor de expresiones faciales. Todas ellas están enfocadas a conseguir que el sistema sea más autónomo y social.

[3] http://www.aldebaran.com/en/humanoid-robot/nao-robot/ (Accedida el 2/9/2014)

1.3.1. Diseñador de terapias

En el hospital, los terapeutas deben transformar manualmente los objetivos terapéuticos en un conjunto de ejercicios distribuidos en diferentes sesiones de rehabilitación que compongan el plan de terapia completo, teniendo en cuenta todos los requisitos establecidos previamente por el médico y evitando que las sesiones sean demasiado parecidas entre sí.

Tener muchos ejercicios diferentes para un mismo objetivo terapéutico es conveniente porque enriquece la calidad de la terapia. Sin embargo, seleccionar los ejercicios adecuados para los objetivos terapéuticos, respetando las restricciones de diseño de la sesión y del paciente mientras se asegura la variabilidad es una tarea que consume mucho tiempo para los terapeutas.

Éstos a menudo no se preocupan por encontrar un conjunto de ejercicios que respete todos estos criterios. Para ahorrar tiempo, los terapeutas improvisan los ejercicios que tendrá cada sesión, de modo que no siempre se ciñen a los objetivos terapéuticos ni se saca provecho de la variedad de ejercicios disponible. Esta dependencia en la intuición del terapeuta puede reducir el compromiso del paciente con la terapia y la efectividad de la misma.

Éste es un problema de búsqueda que puede resolverse utilizando Planificación Automática [Ghallab et al. 2004]. Por ello, NAOTherapist incorpora un diseñador de terapias basado en la planificación clásica que utiliza esta técnica de inteligencia artificial. Esto asegura el cumplimiento de todos los requisitos establecidos por el médico y la variedad de las sesiones. De este modo, el terapeuta humano puede simplemente recibir del sistema una lista con los ejercicios a entrenar, en lugar de tener que improvisarlos cada vez.

Paralelamente, otra persona ha desarrollado un sistema similar basado en el paradigma de planificación jerárquica. La comparación de las características y dificultades de modelado entre la aproximación desarrollada para este trabajo y la jerárquica se ha

publicado en junio de 2014 [Pulido et al. 2014].

1.3.2. Robot terapeuta

La lista de ejercicios que genera el diseñador de terapias puede enviarse a un robot para que haga él mismo la terapia, aumentando muchísimo la autonomía del sistema. Cuando el paciente hiciese una sesión con NAOTherapist, el terapeuta podría limitarse a tareas puntuales de supervisión del robot y de evaluación del progreso de la rehabilitación. Esto no significa que el terapeuta pueda ser completamente sustituido por un robot. NAOTherapist está pensado como una herramienta de apoyo a la terapia cuya autonomía favorece la interacción y puede ahorrar tiempo de especialistas, pero en última instancia será necesario que esté supervisada por un ingeniero.

El uso de un agente físico en terapias de rehabilitación es algo muy innovador sobre lo que se está investigando mucho. La colaboración activa de robots en sesiones de rehabilitación realmente supone un factor de ahorro de tiempo y permite automatizar la supervisión de la terapia y la monitorización. Muchos estudios demuestran que los beneficios de los robots en este tipo de tratamientos son muy significativos [Drużbicki et al. 2013, Ros et al. 2011, Borggraefe et al. 2010]. Los robots suelen ser muy atractivos para los niños, haciendo que la terapia sea menos aburrida y permiten que los niños se impliquen más en la misma [Matarić et al. 2007].

NAOTherapist está enmarcado dentro de la robótica social, que tiene ciertos requisitos específicos. El más obvio es que el agente físico (el robot) tenga un aspecto adecuado para que el niño pueda empatizar con él. En este trabajo se ha podido utilizar el robot NAO, cuyas características físicas hacen que sea un buen dispositivo para desarrollar sistemas de robótica social [Fridin et al. 2011]. Sin embargo, un comportamiento coherente y natural es lo más importante para conseguir que el niño sea capaz de atribuir cualidades inteligentes al robot. Es por ello que se ha desarrollado una arquitectura cognitiva para él.

Aunque la arquitectura desarrollada en este trabajo es ad-hoc para el robot NAO, el uso de RoboComp [Manso et al. 2010] permite que sus componentes puedan reutilizarse fácilmente en otra arquitectura más general. La ejecución del sistema debe estar organizada por un componente Ejecutivo que sea el que recoge la información de los sensores disponibles, pida la siguiente acción del robot a un planificador de nivel medio y finalmente la ejecute en el robot. La implementación de este componente Ejecutivo es otra de las partes de este trabajo.

1.3.3. Reconocedor de expresiones faciales

La cámara del robot NAO no es lo suficientemente sofisticada para reconocer las posturas de los brazos del paciente con precisión ni tampoco para inferir distintas características útiles y necesarias para un robot social (como la expresión de la cara del paciente). Por ello NAOTherapist se apoya en una cámara externa 3D Kinect de Microsoft[4].

La detección de expresiones faciales es una parte de la visión por ordenador que ha sido objeto de mucha investigación en los últimos años. Se engloba dentro del término más general denominado computación afectiva [Zeng et al. 2009]. Los robots sociales tienen que interactuar de forma más directa con los seres humanos por lo que es importante que sean sensibles a las emociones y el humor de las personas. Los sistemas de reconocimiento de expresiones faciales ayudan a desarrollar una interfaz visual coherente entre humano y robot.

De las caras humanas pueden extraerse muchos puntos característicos diferentes que se sitúan por toda la cara: en los ojos, la punta de la nariz, las comisuras de la boca, las orejas, etc. Los sistemas de detección avanzados pueden detectar más de 100 puntos diferentes. Una sonrisa, un grito o una cara seria cambian la distribución de esos puntos de forma distintiva. Modelar a mano un sistema capaz de extraer información de esos puntos es muy difícil ya que las reglas que rigen estos cambios en las posiciones de los

[4] http://www.xbox.com/es-ES/Kinect/ (Accedida el 2/9/2014)

puntos no son evidentes. Por ello, muchas técnicas de reconocimiento de expresiones faciales utilizan técnicas de Aprendizaje Automático [Bettadapura 2012].

Cambiar de una expresión facial a otra no modifica un único punto individual, sino que influye en todo un conjunto de ellos. Es decir, los puntos están relacionados entre sí de algún modo, aunque el mecanismo sea difícil de determinar. Estas relaciones pueden modelarse con lenguajes relacionales [Dzeroski 2005] para crear un modelo que permita clasificar las expresiones faciales. Otros métodos de Aprendizaje Automático más tradicionales de tipo atributo-valor solo tienen en cuenta valores individuales de las variables, pero no exploran las relaciones que podría haber entre ellas. Por eso en este trabajo se ha utilizado el algoritmo Tilde incluido en el software de minería de datos relacional ACE[5].

A pesar de que existe mucha literatura sobre clasificación de expresiones faciales, el uso de modelos relacionales es poco frecuente en este ámbito. A fecha de escritura, el autor no ha encontrado ningún otro intento de clasificar expresiones faciales mediante minería de datos relacional. Por todo esto se decidió investigar esta técnica para evaluar qué resultados ofrece y especialmente ver qué se puede aprender de las relaciones entre variables extraídas.

1.4. Objetivos del trabajo

Los objetivos concretos de este trabajo se pueden resumir del siguiente modo:

- Diseñar una arquitectura cognitiva para NAOTherapist que sea fácilmente generalizable al proyecto THERAPIST. La carga de trabajo de esta parte ha sido del 50 % por cada miembro del equipo de desarrollo.
- Utilizar la Planificación Automática clásica para modelar e implementar en la arquitectura de NAOTherapist un nuevo dominio de planificación capaz de diseñar las terapias de rehabilitación de los pacientes automáticamente. Como guía clíni-

[5] http://dtai.cs.kuleuven.be/ACE/ (Accedida el 2/9/2014)

ca para el modelado se siguen los protocolos médicos del Hospital Universitario Virgen del Rocío.

- Implementar el componente Ejecutivo de la arquitectura, que sirve de nexo de unión entre todos los componentes y controla al robot según sea la siguiente acción que le indique el planificador de nivel medio.
- Investigar el uso de la minería de datos relacional en el reconocimiento de expresiones faciales para, en una fase posterior, mejorar la interacción social de NAOTherapist y el proyecto THERAPIST.

1.5. Estructura de la memoria

La memoria está estructurada de forma que primero se va explicando lo más general y poco a poco va centrándose en lo específico. El capítulo actual corresponde a esta introducción y motivación general. El Capítulo 2 presenta el estado de la cuestión, que incluye subsecciones teóricas y técnicas sobre las soluciones que se han adoptado en este trabajo. El Capítulo 3 explica la arquitectura NAOTherapist y detalla la implementación de las partes principales de este trabajo: el diseño de terapias con Planificación Automática clásica, el componente Ejecutivo y el reconocedor de expresiones faciales. En el Capítulo 4 se encuentran todos los experimentos con los que se evalúa el trabajo. Finalmente, en el Capítulo 5 se discuten las conclusiones y el trabajo futuro de todo lo explicado anteriormente.

A pesar de que, por claridad, se ha decidido separar la evaluación de la descripción de lo que se ha hecho, el lector puede decidir otro orden de lectura. Se pueden leer los capítulos en este orden: 3.1, 4.1, 3.2, 4.2, 3.3 y 4.3. De este modo se explican juntas la implementación, experimentación y conclusiones de cada parte del trabajo.

Por último, dada la gran cantidad de detalles técnicos de esta memoria, se han añadido varias notas al pie con sitios web para conceptos que no se podían referenciar de otro modo, particularmente de empresas y organizaciones privadas.

Capítulo 2

Estado de la cuestión

Este capítulo ofrece una panorámica general de las diferentes aproximaciones que ha habido en la literatura sobre las principales partes de este trabajo. También se detallan diversos conceptos teóricos importantes en los que se apoyan las soluciones desarrolladas.

La primera sección explica diversos trabajos modernos sobre robótica social de asistencia, se aborda el proyecto THERAPIST, el robot NAO y el *framework* robótico en el que se basa la arquitectura NAOTherapist. La segunda parte trata el diseñador de terapias, la Planificación Automática clásica y el *framework* de planificación PELEA. Finalmente se explica el reconocimiento de expresiones faciales, la minería de datos relacional y la manera en la que la cámara Kinect captura los rasgos de la cara.

2.1. Robótica social de asistencia

El desarrollo de dispositivos para recuperar funciones neurológicas es un gran reto para muchos médicos e ingenieros [Tapus et al. 2007, Nalin et al. 2012]. En concreto las aplicaciones de robótica social han crecido mucho durante la última década. Este concepto representa a todos los robots que proveen algún tipo de servicio de asistencia

a la gente. Para este trabajo resulta especialmente interesante la taxonomía de robots de Feil-Seifer y Mataric [Feil-seifer et al. 2005], ya que tiene en cuenta explícitamente los robots de asistencia. Ésta engloba tres categorías principales resumidas en la Figura 2.1:

- SIR (*Socially Interactive Robotics*): Los robots socialmente interactivos son aquellos cuya tarea principal es algún tipo de interacción social, aunque no tienen necesariamente que tener una finalidad de asistencia al usuario. Los robots mayordomo [Graf et al. 2009] y los robots de entretenimiento son un buen ejemplo.
- AR (*Assistive Robotics*): Engloba a todos los robots que proveen algún tipo de asistencia a la gente, aunque sea sin interacción social. Un ejemplo serían los exoesqueletos vestibles para usuarios con daños en la espina dorsal que sirven para facilitar acciones cotidianas como caminar o subir escaleras [Rosen et al. 2007]. También incluye robots de terapia. Unos se adaptan a una parte del cuerpo del usuario (como un brazo) para obligarle a hacer ciertos movimientos de rehabilitación que normalmente no puede hacer. Otros permiten que pacientes con más capacidad motora puedan entrenarse con robots que ejercen cierta resistencia al movimiento [Burgar et al. 2000, Kahn et al. 2001].
- SAR (*Socially Assistive Robotics*): Esta categoría es la intersección entre AR y SIR. Incluye robots que proveen algún tipo de asistencia únicamente mediante interacción social, sin contacto físico. NAOTherapist se engloba dentro de esta categoría. Precisamente, las terapias en las que un terapeuta entrena una serie de ejercicios con un paciente han demostrado ser uno de los métodos de rehabilitación más efectivos. Por este motivo las aproximaciones SAR sin contacto físico entre paciente y robot autónomo que asiste como terapeuta son las que están en la línea de investigación actual [Eriksson et al. 2005].

Probar y desplegar una plataforma SAR es relativamente fácil porque reduce el riesgo de seguridad para el paciente al no necesitar contacto físico. Los pacientes son capaces de establecer lazos emocionales con estos robots, mejorando su motivación

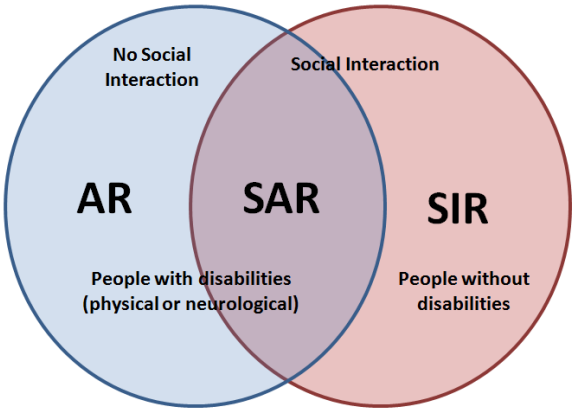

Figura 2.1: Relaciones entre *Assistive Robotics*, *Socially Interactive Robotics*, y *Socially Assistive Robotics*.

para seguir el tratamiento [Matarić et al. 2007]. La comunicación verbal y no verbal, la voz, gestos y la apariencia física son una parte muy importante de la interacción humano-robot. Trasladar el conocimiento del afecto humano a sistemas SAR los hace más efectivos, naturales y parecidos a la experiencia social humana [Tapus et al. 2006].

Idealmente, una plataforma SAR debe ser capaz de moverse de forma autónoma en entornos humanos, interactuando y socializando con la gente. Esto requiere la investigación y el desarrollo de técnicas de soporte para habilidades y comportamientos completamente nuevos para conseguir que estos robots se comuniquen y comprendan a los seres humanos. Entre estas técnicas se incluye la detección de expresiones faciales.

Existen varias aproximaciones SAR con distinto nivel de éxito y sofisticación. Las líneas de trabajo a veces usan juegos, técnicas de virtualización y realidad aumentada para mejorar la experiencia interactiva del paciente [McMurrough et al. 2012].

La versatilidad y el bajo precio de los robots Pioneer2 DX provoca que se utilicen para todo tipo de aplicaciones, incluida la realización de terapias, aunque carezcan de un aspecto adecuado para la interacción social. El sistema propuesto por Mataric [Eriksson et al. 2005, Matarić et al. 2007] está dirigido a pacientes ancianos que han tenido un ictus y perdieron movilidad en las extremidades superiores. El robot sigue

al paciente detectando sus piernas con un láser y le propone verbalmente juegos y ejercicios de rehabilitación motora a lo largo de un circuito. Dispone de una cámara para detectar las reacciones en tiempo real y actuar en consecuencia. La interacción se basa en mensajes pregrabados.

Fasola utiliza un robot humanoide de un metro de alto que también sirve como terapeuta o entrenador para ancianos [Fasola et al. 2010]. En este caso no se aplican criterios clínicos precisos, por lo que en todo momento el entrenamiento se propone como un juego. Los autores se centran en la interacción social para analizar si los usuarios llegan a percibir al robot como una entidad inteligente. En esencia, el robot realiza movimientos con los brazos y propone varios juegos para que los usuarios los repitan y estimulen otras habilidades cognitivas, como la memoria.

En 2013, Choe utiliza criterios médicos estrictos para realizar una evaluación clínica mucho más precisa de un paciente que ha terminado una terapia robótica completa [Choe et al. 2013]. El robot es un uBot-5 con brazos y una pantalla en la cabeza para mostrar caras y otra información que favorezca la interacción social. Propone juegos y ejercicios de rehabilitación de extremidades superiores y otras habilidades cognitivas. Los autores llegan a la conclusión de que la terapia robótica fue más estimulante de lo que hubiera sido otra más tradicional.

Fridin también se basa en el robot NAO para realizar terapias infantiles con niños autistas [Fridin et al. 2011]. En este trabajo se propone una arquitectura cognitiva bastante sofisticada, aunque no se llega a implementar y los autores no hablan de las posibilidades de reutilización de sus componentes. Sí llegan a desarrollar un prototipo en el que el robot mueve los brazos delante de una fila de niños para entrenar con ellos de forma secuencial. El objetivo de los autores se centra en analizar cómo el niño acepta la autoridad del robot y crea lazos afectivos con él.

2.1.1. Proyecto THERAPIST

THERAPIST [Calderita et al. 2013] es un ambicioso proyecto de robótica social que aún está en las primeras fases de desarrollo. Tiene una parte de investigación de la cual NAOTherapist formará parte, además de servir como primer prototipo y prueba de concepto.

Los pacientes objetivo son los mismos que en NAOTherapist, aunque el robot que se usará en la versión final es una evolución de Ursus [Suárez Mejías et al. 2013], un robot humanoide con forma de oso (Figura 2.2). También hará uso de la cámara 3D Kinect de Microsoft. Utiliza el *framework* robótico RoboComp[1], basado en componentes reutilizables. La arquitectura de NAOTherapist también está basada en el *framework* RoboComp para que los nuevos componentes desarrollados se puedan reutilizar fácilmente en THERAPIST.

Figura 2.2: El robot Ursus (izquierda) y su evolución para el proyecto THERAPIST (derecha).

En THERAPIST y Ursus, el protocolo clínico tiene dos fases. En la primera, una pantalla muestra el vídeo de una persona realizando un ejercicio. Después, el robot reproduce el mismo ejercicio con sus brazos y le pide al paciente que le imite. Si el

[1] http://robocomp.sourceforge.net/ (Accedida el 2/9/2014)

paciente no lo hace correctamente, el robot es teleoperado para que le muestre cómo hacerlo correctamente. En una segunda fase, el paciente juega con una serie de juegos de realidad aumentada que le fuerzan a entrenar los movimientos terapéuticos.

La fase experimental de Ursus mostró que los pacientes disfrutaron del robot en las sesiones de rehabilitación. Sin embargo el robot no era suficientemente autónomo ya que estaba programado como una máquina de estados y no permitía replanificar la sesión ni aprender nuevas formas de interacción. Este sistema no permitía ahorrar tiempo de especialistas, por lo que se decidió expandirlo con el proyecto THERAPIST. El objetivo principal de THERAPIST es dotar de mayor autonomía y capacidad social al robot usado en Ursus mediante la inclusión de técnicas de inteligencia artificial como la Planificación Automática y el Aprendizaje Automático. Al incrementar la autonomía y la interacción social es necesario investigar e incluir técnicas que las sirvan de base y permitan su aplicación. Así se podrá extraer más información de los sensores para que la planificación del comportamiento sea más coherente.

Al igual que en NAOTherapist, el objetivo no es sustituir completamente al terapeuta, ya que el robot es una herramienta de apoyo a la terapia que, aunque le ahorre tiempo gracias a su autonomía, necesita supervisión.

2.1.2. NAO

NAOTherapist utiliza el robot humanoide NAO H21 de Aldebaran (Figura 2.3). Aunque el robot dispone de sensores como sónar, cámaras 2D y micrófonos, éstos no son suficientemente sofisticados para capturar los movimientos de brazos del paciente y determinar si realiza bien el ejercicio o no. Para este fin se utiliza una cámara Kinect externa.

El NAO, sin embargo, tiene varias características que lo hacen especialmente útil en este caso. Dispone de dos altavoces y un sistema para sintetizar voz a partir de texto en inglés. Su aspecto divertido gusta mucho a los niños, por lo que facilita que

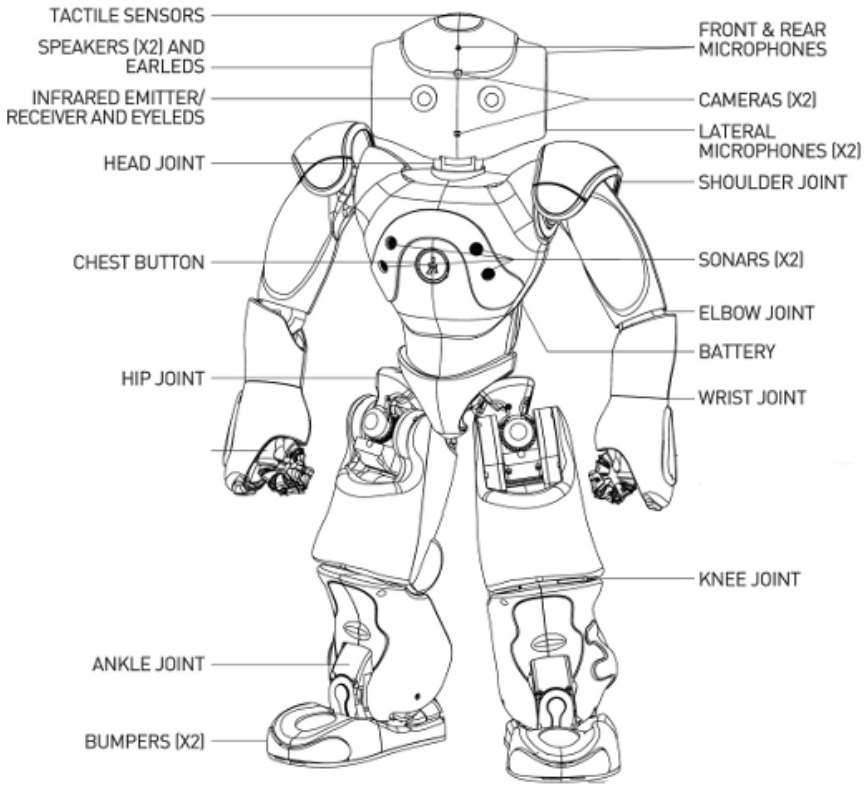

Figura 2.3: Partes del robot NAO H21.

puedan empatizar con él. Además, cada brazo tiene seis grados de libertad: dos en el hombro, dos en el codo, uno para rotar la muñeca y otro que cierra y abre la mano. En este trabajo la muñeca y la mano no son relevantes, ya que la cámara Kinect no es capaz de reconocer estas articulaciones en el paciente. Sin embargo sí es capaz de leer las otras cuatro (Figura 2.4). Con esas cuatro articulaciones el NAO puede hacer una gran variedad de posturas con cada brazo, por lo que resulta muy adecuado para el prototipo de NAOTherapist.

El robot no puede hacer todos los procesamientos de NAOTherapist por sí solo, por lo que necesita al menos un ordenador externo. Las comunicaciones las realiza con el componente Ejecutivo que se desarrolla en este trabajo mediante una conexión WiFi. En realidad, el NAO también es en sí mismo un ordenador con una versión muy ligera de Linux. Al iniciarse, arranca diversos componentes que se quedan esperando

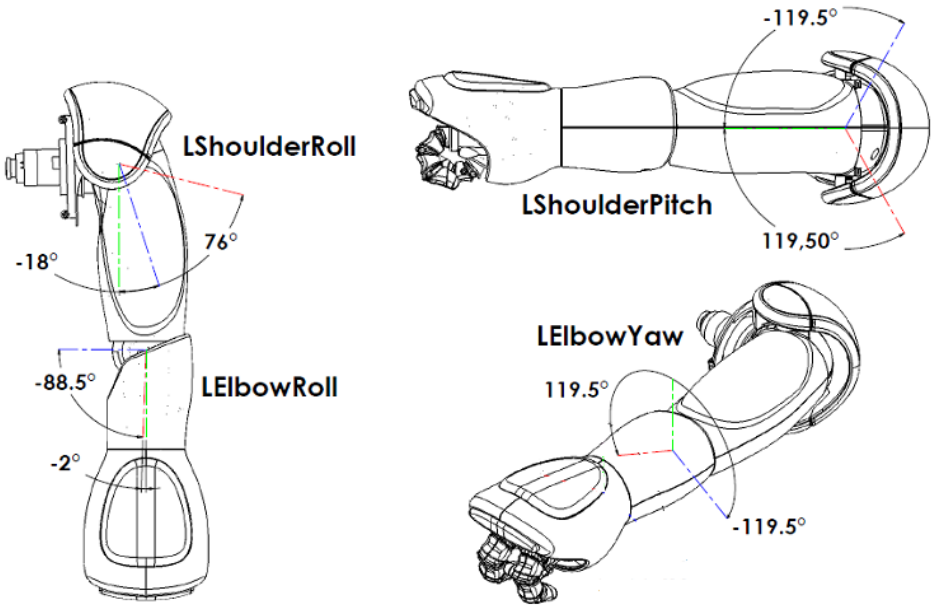

Figura 2.4: Articulaciones relevantes de un brazo del robot NAO H21.

a que otro programa requiera los servicios que ofrecen. El conjunto de estas piezas de software se llama NaoQi[2] y el componente Ejecutivo de NAOTherapist debe interactuar con ellas. Cada componente de NaoQi se encarga de una parte de la funcionalidad del NAO, siendo estos los más importantes para este trabajo:

- AlMotion: Controla el movimiento del NAO, haciendo uso de un planificador de caminos de bajo nivel situado en el propio robot.
- ALRobotPosture: Mueve el robot a una postura preprogramada usando una combinación de movimientos compleja (sentarse, levantarse, etc.).
- ALTextToSpeech: Sintetiza voz que dice cualquier mensaje de texto en inglés.
- ALLeds: Controla el brillo y el color de todas las luces del robot.

NaoQi está específicamente pensado para utilizarse con el lenguaje Python, aunque también es posible usar C++ y Java. El módulo Ejecutivo se ha programado en Python para facilitar la interacción con el robot.

[2] https://community.aldebaran-robotics.com/doc/1-14/naoqi/index.html (Accedida el 2/9/2014)

2.1.3. RoboComp

RoboComp [Manso et al. 2010] es el *framework* robótico utilizado en Ursus y el que también se utiliza en THERAPIST. También se usa en NAOTherapist para que la arquitectura desarrollada en este trabajo sea más fácilmente generalizable y reutilizable. RoboComp está formado por diferentes componentes que son piezas de software individuales, similar a NaoQi dentro del propio robot NAO.

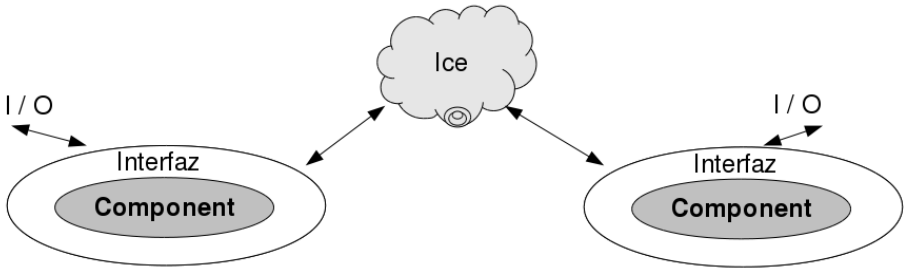

Figura 2.5: Representación genérica de componentes RoboComp.

Sin embargo, RoboComp está pensado para que cada componente pueda ser distribuido en diferentes máquinas. Las comunicaciones entre componentes se realizan por medio del *middleware* ICE[3] (*Internet Communications Engine*). Los componentes únicamente tienen que implementar una interfaz ICE que interactúe con este *middleware* y éste les facilita una comunicación basada en eventos mediante TCP/IP. Esto es muy útil ya que existen componentes que funcionan mejor en ciertos sistemas operativos (por ejemplo, los planificadores y controladores hardware suelen funcionar mejor en Linux y otros, como la Kinect de Microsoft, mejor en Windows). De hecho, el prototipo de NAOTherapist funciona simultáneamente sobre dos máquinas diferentes: una con Windows dedicada al componente RoboComp de la Kinect llamado WinKinectComp y otra con Linux para el resto de componentes.

Además, también pueden usarse componentes programados en distintos lenguajes

[3] http://www.zeroc.com/overview.html (Accedida el 2/9/2014)

de programación. NAOTherapist usa componentes programados en Java, Python y C sharp. Los nuevos componentes RoboComp que ofrezcan algún servicio tienen que compartir la definición de su interfaz de comunicación (un fichero .ice) para que otros componentes sepan cómo reutilizarlos en el futuro.

Además de los nuevos módulos RoboComp programados para la arquitectura de NAOTherapist, se hace uso de otros módulos del *framework* ya implementados.

- PELEAComp: Sirve para controlar la planificación de las sesiones y replanificar el plan cuando una acción no ha tenido el efecto esperado.
- WinKinectComp: Hace uso del software oficial de Microsoft para capturar el esqueleto y la cara del paciente.

En comparación con otros *frameworks* robóticos, RoboComp intenta solucionar los problemas de escalabilidad, flexibilidad y reusabilidad que suelen surgir en robótica mediante la programación orientada a componentes. Esta aproximación es especialmente útil en robótica debido a lo heterogéneo que es el software que se utiliza en ella. Normalmente la reutilización de componentes en este campo es muy difícil porque los componentes tradicionales suelen ser poco generalizables. Las arquitecturas basadas en RoboComp facilitan esta generalización y la reutilización.

2.2. Generación de terapias

Los Sistemas de Soporte a la Decisión Clínica (SSDC) se han ido desarrollando en las últimas décadas para facilitar muchas tareas de los médicos. En algunos casos, el protocolo para tratar a un paciente puede ser confuso por depender de muchas variables y de un conjunto de objetivos terapéuticos que el paciente debe alcanzar. A menudo, los médicos solo disponen de pautas clínicas con recomendaciones de muy alto nivel sobre qué combinación de tratamientos se pueden establecer para las enfermedades e interacciones que puede tener un paciente. A menudo, el equipo médico debe reunirse para discutir la combinación de pasos clínicos que minimice los riesgos para el paciente

y maximice el resultado esperado, por ejemplo, de acuerdo a una escala estandarizada [Peleg 2013].

Respecto a la generación de planes de terapia o tratamientos hay algunos trabajos en la literatura. Ahmed et al. presenta un sistema para la generación automática de tratamientos en pacientes de cáncer [Ahmed et al. 2010]. El sistema maneja la correcta selección de la geometría e intensidad de las irradiaciones para optimizar la dosificación del tratamiento. En 2010, Morignot también usa Planificación Automática para generar escenarios que ayuden a personas discapacitadas [Morignot et al. 2010]. González-Ferrer usa un algoritmo de planificación capaz de generar planes de tratamiento oncológicos [González-Ferrer et al. 2013, Fdez-Olivares et al. 2011], transformando pautas interpretables por ordenador de la enfermedad de Hodgkin, incluyendo restricciones temporales difíciles de planificar por los médicos. Schimmelpfeng presenta una aproximación con programación Entera-Mixta para determinar citas de pacientes en hospitales de rehabilitación [Schimmelpfeng et al. 2012], aunque no planifica ejercicios dentro de cada sesión para alcanzar unas metas determinadas de acuerdo a los requisitos del paciente.

2.2.1. Planificación Automática

La Planificación Automática es la rama de la Inteligencia Artificial que se ocupa de dotar a las máquinas con la capacidad de hacer planes que consigan objetivos [Ghallab et al. 2004]. Más formalmente, la Planificación Automática es el proceso de encontrar una secuencia ordenada de acciones que, partiendo de un estado inicial, lleguen a otro estado en el que se cumplan una serie de metas. Generalmente las acciones se expresan en términos de precondiciones que se deben cumplir para que la acción se pueda aplicar, y de efectos que son los cambios que se producen en el estado del mundo al aplicar la acción. La Planificación Automática independiente del dominio utiliza técnicas genéricas de resolución de problemas para encontrar esta secuencia de acciones independientemente de cuáles sean las acciones, las metas y el problema en el

que se apliquen.

Los planificadores automáticos son las piezas de software que implementan técnicas de Planificación Automática para encontrar planes de acciones [Hoffmann 2003, Fuentetaja et al. 2010, Benton et al. 2012]. Habitualmente utilizan un fichero de definición de dominio y otro de problema escritos en lenguaje PDDL (*Planning Domain Definition Language*) [Fox et al. 2003]. El dominio contiene la definición de los predicados, funciones y tipos que con los que se modelan los objetos y las relaciones del mundo que se considera en el problema de planificación. También contiene las posibles acciones del plan, con sus precondiciones y efectos que el planificador tendrá que manejar para encontrar un plan válido. El problema contiene el estado inicial, la definición de objetos y las metas.

La Planificación Automática clásica se caracteriza por tener un único estado inicial bien definido, sobre el cual un único agente realiza acciones deterministas y sin duración que pueden ejecutarse de una en una. En la versión 2.1 de PDDL se añadió la posibilidad de trabajar con variables numéricas (denominadas fluents) y con costes de acción. Los costes de acción permiten optimizar un plan no solo reduciendo el número de acciones, sino ejecutando aquellas que tengan menor coste asociado. Las variables numéricas dotan de mucha mayor expresividad al lenguaje PDDL gracias a las precondiciones numéricas. Estas dos características son bastante nuevas y no la soportan todos los planificadores, pero en la práctica son extremadamente útiles a la hora de modelar.

2.2.2. PELEA

En la actualidad, la mayoría de las técnicas de control de robots adolecen de falta de capacidad para cambiar su comportamiento ante eventos inesperados o de realizar acciones complejas no programadas con anterioridad, lo cual puede ser resuelto mediante el uso de la Planificación Automática. Las dificultades técnicas inherentes a la Planificación Automática y la robótica han hecho que durante casi tres décadas estas áreas se hayan desarrollado sin casi interacción.

Figura 2.6: Diagrama de funcionamiento básico de PELEA.

PELEA (*Planning, Execution and Learning Architecture*) [Alcázar et al. 2010] es una arquitectura genérica para la integración de planificación, ejecución, monitorización, replanificación y aprendizaje. Está específicamente pensada para utilizarse en robótica, aunque también sirve en otros ámbitos. El diseño de PELEA es completamente modular, de forma que es posible personalizarla para añadir o quitar módulos según sea necesario. Está integrada dentro del *framework* de RoboComp, por lo que se ha utilizado de forma directa en NAOTherapist.

Aunque el diseño canónico de PELEA tiene muchos módulos diferentes, solo tres de ellos son relevantes en NAOTherapist (Figura 2.6). La comunicación entre PELEA y el resto de la arquitectura la lleva a cabo el módulo Execution (no confundir con el componente Ejecutivo de NAOTherapist). En esencia, PELEA se encarga de planificar un plan de acciones e ir devolviendo una a una las acciones del mismo. Cada acción tiene una serie de efectos que deben cumplirse en el estado del mundo. Si a PELEA le llega un estado que no se corresponde con el esperado, replanifica y genera otro plan de acciones.

El funcionamiento básico es el siguiente: Execution recibe la petición del componente Ejecutivo con el estado del mundo actual y se lo manda a Monitoring. Si el estado recibido y el esperado son iguales, Monitoring busca en la variable infoMonitoring la siguiente acción previamente planificada y se la manda a Execution. Si el estado actual es diferente del esperado, el plan previo ya no es válido, por lo que Monitoring pide un nuevo plan a Decision support. Decision support llama a un planificador o replanificador automático externo e interpreta el plan que éste le genera. Una vez interpretado,

se lo envía a Monitoring para que lo almacene en su variable infoMonitoring. Esta variable evita tener que planificar cada vez que se le pide una acción a PELEA, lo que aumentaría mucho el coste computacional. Finalmente Monitoring envía la siguiente acción a Execution. En la arquitectura NAOTherapist, el módulo Execution de PELEA y el componente RoboComp Ejecutivo están conectados entre sí, tal como se muestra en la Figura 3.1.

2.3. Reconocimiento de expresiones faciales

Existen casos reales en los que se han utilizado algunos algoritmos de reconocimiento de expresiones faciales. Se pueden ver en cámaras de fotos con disparo automático cuando detectan sonrisas, en videojuegos con avatares que imitan la expresión del usuario o en salas de chat donde se insertan emoticonos automáticamente dependiendo del estado de ánimo detectado. También podría ser útil para otros campos como el de las telecomunicaciones, ciencia del comportamiento, videojuegos, animaciones, psiquiatría, seguridad en el automóvil, televisores, equipos de sonido sensibles al humor del usuario, software educativo, etc.

En la literatura científica, se pueden encontrar varios artículos [Bettadapura 2012, Sumathi et al. 2012] con estados de la cuestión actuales de clasificadores de expresiones faciales.

El sistema desarrollado está centrado en un clasificador que utiliza el algoritmo de aprendizaje relacional Tilde. La mayoría de las aproximaciones utilizan puntos o vectores característicos que se adaptan a las facciones de una cara. En este trabajo se ha utilizado la malla de puntos que proporciona el propio software de Microsoft. Sin embargo, en la literatura, los autores suelen desarrollar paralelamente su propio sistema de detección y extracción de características adaptado al tipo de clasificador que pretenden utilizar.

Algunos también utilizan más de un fotograma para determinar la pose y la expre-

sión de la cara, de forma que pueden aprovechar información temporal. Varios trabajos que explotan esta temporalidad se basan, por ejemplo, en reglas difusas con algoritmos genéticos [Jamshidnezhad et al. 2011] o clasificadores AdaBoost [Yang et al. 2009]. Es cierto que la temporalidad puede dar pistas sobre la expresión facial que se tiene en un cierto momento, sin embargo las expresiones pueden cambiar de forma repentina en caso de emergencia. Además, los errores de detección debidos a oclusiones de la imagen (poner una mano sobre la cara, etc.) pueden arrastrarse entre fotogramas. Por este motivo existen aproximaciones que solo utilizan la información de una única imagen. Ejemplos de este tipo utilizan la reducción de dimensionalidad (PCA) con redes de neuronas artificiales [Thai et al. 2011] o AdaBoost con SVMs [Bartlett et al. 2006]. No se ha encontrado ninguno que utilice el algoritmo de aprendizaje relacional Tilde.

La precisión de la clasificación de los sistemas revisados varía entre el 80 % y el 90 %. Sin embargo es difícil compararlos entre ellos debido a las grandes diferencias en la metodología, los ejemplos utilizados y el número de expresiones que pueden clasificar.

2.3.1. Aprendizaje Automático relacional

Para deducir qué expresión tiene una cara en función de los puntos anteriormente mencionados se utiliza el sistema de minería de datos ACE[4], que provee una interfaz común para diferentes algoritmos de minería de datos relacional. La minería de datos relacional es el proceso de encontrar patrones en una base de conocimiento en la que distintos elementos, quizá en distintas tablas, tienen relaciones entre ellos. Extiende la minería de datos clásica en el sentido de que en ella solo se encuentran patrones dentro de filas individuales, mientras que los encontrados por la minería de datos relacional se pueden extender a lo largo de diferentes filas, tablas y relaciones.

Tilde es uno de los algoritmos que soporta este software y es el que se ha utilizado en este trabajo. Este algoritmo es una ampliación del árbol de decisión C4.5 hacia la minería de datos relacional. Construye árboles de decisión que describen relaciones

[4] http://dtai.cs.kuleuven.be/ACE/ (Accedida el 2/9/2014)

entre elementos de la base de conocimiento. Estos árboles son útiles no solamente porque sean capaces de clasificar y predecir bien, sino porque de su lectura es posible extraer conclusiones interesantes de la base de conocimiento que pueden no ser nada evidentes a simple vista.

De forma básica, ACE lee la base de conocimiento de un fichero .kb (predicados basados en puntos característicos de la cara de varios ejemplos) y un fichero .s que describe el modelo de la base de conocimiento y la configuración que regirá el aprendizaje. El programa ejecuta el algoritmo de Tilde sobre los ejemplos de entrenamiento y extrae un árbol de decisión relacional. Para clasificar más instancias basta con leer este árbol y realizar las consultas Prolog equivalentes o ejecutando la transcripción a código Prolog que exporta ACE.

2.3.2. Kinect

En 2010 Microsoft lanzó al mercado la cámara Kinect[5] como un periférico para su videoconsola Xbox 360. La idea fundamental era que los jugadores pudiesen interactuar con los videojuegos usando solo su cuerpo, sin necesidad de mandos externos. La cámara dispone de un sensor RGB, un sensor infrarrojo y un sensor de profundidad. El bajo coste, su conexión USB y el soporte oficial para Windows y no oficial para Linux provocaron que pronto se convirtiese en el sensor de profundidad más utilizado en el ámbito universitario para hacer investigaciones de visión por ordenador.

El Face Tracking SDK[6] de Microsoft se conecta a una cámara Kinect y permite desarrollar aplicaciones que pueden seguir caras humanas en tiempo real. Para ello primero es necesario detectar el esqueleto de una persona sentada, como se puede ver en la Figura 2.7a. Al contrario que en otras aplicaciones de Kinect, en este trabajo no es necesario detectar el esqueleto completo. Después, el software busca en la zona de la cabeza para encontrar los puntos característicos de la cara en tiempo real. Se pueden

[5] http://www.xbox.com/es-ES/Kinect/ (Accedida el 2/9/2014)
[6] http://msdn.microsoft.com/en-us/library/jj130970.aspx (Accedida el 2/9/2014)

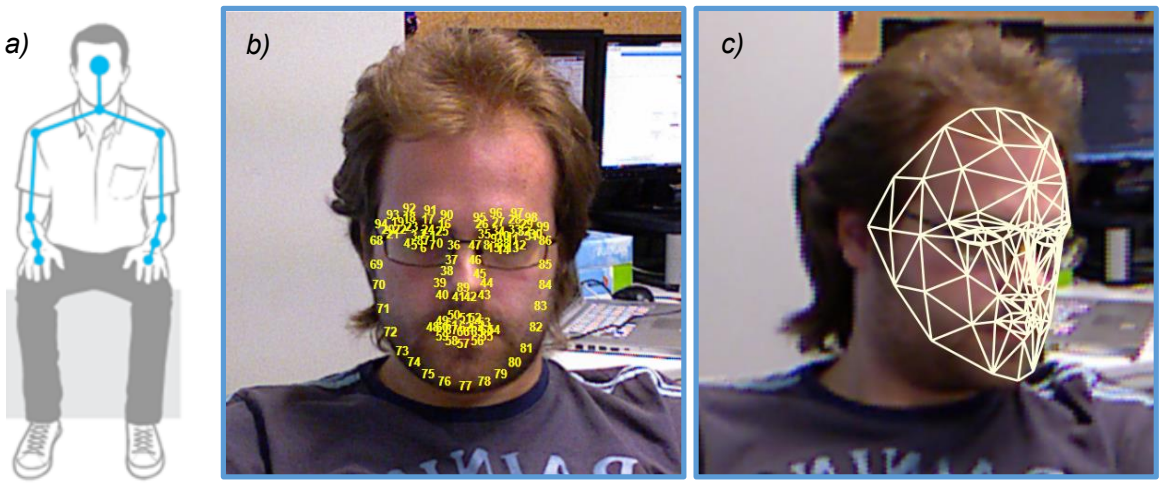

Figura 2.7: a) Esqueleto sentado. b) Malla de Forma. c) Malla de Modelo.

extraer dos mallas diferentes de puntos en 2D, dependiendo del uso que se le quiera dar.

- **Malla de Forma** (Figura 2.7b): 100 puntos básicos que cubren detalladamente el perfil de la cara, la boca, la nariz, los ojos y las cejas.
- **Malla de Modelo** (Figura 2.7c): 121 puntos que corresponden con los vértices del modelo de cara genérico Candide-3[7]. Este modelo se ajusta a las facciones particulares de la persona y después se proyecta en 2D. Tiene menos puntos para cejas, boca y ojos, pero abarca zonas de la frente y las orejas que no tiene en cuenta la Malla de Forma.

El propio SDK también sugiere una selección de 70 puntos característicos de la Malla de Modelo. Sea cual sea el conjunto de puntos que se utilice, con ellos es posible recrear una cara artificial (tal como se hace con la Malla de Modelo) para, por ejemplo, hacer que un avatar virtual imite las expresiones faciales de una persona. También puede analizarse para extraer conclusiones, como determinar hacia dónde está mirando la persona, si sonríe, si tiene la boca abierta, etc. Por ello estos puntos característicos se usan en muchos de los trabajos del estado de la cuestión.

[7] http://www.icg.isy.liu.se/candide/ (Accedida el 2/9/2014)

A pesar de que las herramientas que aporta Microsoft permiten un desarrollo muy rápido de las aplicaciones, el SDK utilizado tiene diversas limitaciones:

- No es capaz de adaptar la malla de puntos de la boca para expresiones de tristeza (comisuras de los labios bajadas).
- No detecta la cara si está demasiado cerca de la cámara (menos de 0,5 m).
- No detecta la cara si está demasiado de perfil (si rota más de unos $\pm 70^{o}$ respecto del objetivo).

Capítulo 3

Arquitectura NAOTherapist

Para este trabajo, el equipo de desarrollo de NAOTherapist (el autor y José Carlos Pulido), ha diseñado desde cero una arquitectura robótica ad-hoc para el robot NAO. En su desarrollo se han tenido que compatibilizar dos objetivos: primero, que sus componentes fuesen fáciles de trasladar a la futura arquitectura del proyecto THERAPIST, y segundo, que permitiese un desarrollo rápido para disponer pronto de un primer prototipo funcional. Antes de explicarla completamente se comenta el escenario completo para el que se ha diseñado la arquitectura entera de NAOTherapist. El prototipo desarrollado en este trabajo tiene algunas diferencias respecto al siguiente escenario que se explican en la Sección 4.4.

Escenario: El médico diagnostica a un paciente y le receta una terapia robótica de rehabilitación. Configura el diseñador de terapias para que planifique la terapia dependiendo de sus criterios médicos. El paciente deberá acudir al centro donde esté el robot para hacer sus ejercicios de rehabilitación. Antes de empezar la sesión, el terapeuta debe analizar si es necesario algún ejercicio nuevo y, en caso necesario, lo introduce en la base de conocimiento.

Cuando el paciente acude a su sesión de rehabilitación, entra en una sala con el robot NAO y una cámara Kinect. El robot le saluda, le presenta los ejercicios que

van a hacer y hace él mismo cada postura necesaria. El paciente debe imitar al robot, manteniendo la postura tanto tiempo como lo haga el robot, etc. El sistema captura los movimientos de los brazos del paciente con la cámara para ver si realiza los ejercicios correctamente o no. En caso de que no lo haga, el robot le anima a que lo corrija y se esfuerce más. Si el paciente falla demasiado, el robot puede omitir el ejercicio o avisar al terapeuta. Una vez que se han realizado todos los ejercicios, el robot se despide, el paciente se marcha y se marca esa sesión como realizada. En caso de que surja alguna situación inesperada, como que el paciente se marche antes de tiempo, la sesión se pausa y el terapeuta va a comprobar si puede reanudarse o no. Él se encarga de reanudarla si procede.

Tras las sesiones, el terapeuta evalúa el progreso del paciente y actualiza su perfil. Cada ciertas sesiones, el terapeuta envía un informe de progreso del paciente al médico. Con este informe, si el progreso del paciente no es bueno, el médico puede iniciar de nuevo el diseñador de terapias para cambiar la configuración de la terapia y replanificar las sesiones restantes.

A continuación se explican todos los componentes de la arquitectura.

Componentes: En la Figura 3.1 se puede ver el diseño completo. De esta arquitectura, solo se va a profundizar en las partes desarrolladas para este trabajo.

La arquitectura NAOTherapist se ejecuta simultáneamente en dos máquinas diferentes por dos motivos: por un lado, los drivers oficiales de la cámara Kinect solo funcionan en Windows, mientras que los planificadores funcionan en Linux. En cualquier caso, en THERAPIST, también se espera que la arquitectura se ejecute en más de una máquina a la vez, ya que es una de las principales funcionalidades de los componentes RoboComp. La puesta en marcha es muy simple, basta con ejecutar un *script* de *shell* en cada máquina para que arranque los componentes necesarios.

Se cuenta con una base de conocimiento en XML que almacena datos sobre los ejercicios y sobre los pacientes. Los módulos que necesitan obtener información de los ejercicios o de los pacientes pueden leerla. Estos son los ficheros de los que está com-

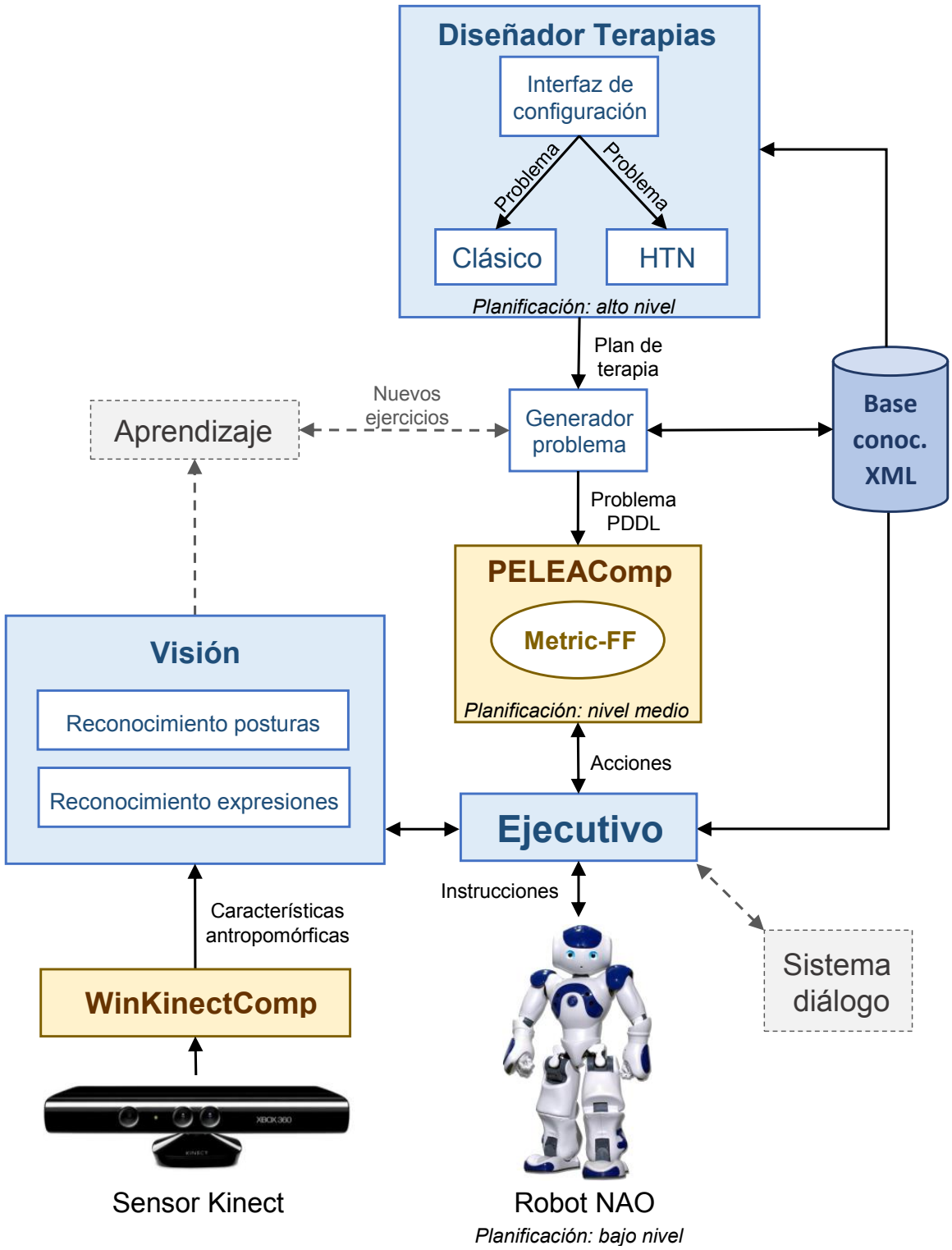

Figura 3.1: Diseño general de la arquitectura de NAOTherapist.

puesta:

- El fichero chains.xml contiene las posturas que el NAO puede adoptar con un brazo. Cada una consiste en un identificador y un ángulo para cada articulación (en total 5). De los 6 grados de libertad de cada brazo solo se tienen en cuenta los dos del hombro y los dos del codo para comparar la postura con la del paciente. Para mejorar la naturalidad de las poses del robot, se ha añadido también la rotación de la muñeca, pero no se usa para comprobar si el paciente ejercita bien o mal porque la cámara Kinect no es capaz de captar con precisión esa rotación.
- El fichero exercises.xml es el que define cada ejercicio. Contiene información general sobre ellos, como los objetivos terapéuticos para los que es útil y los movimientos concretos que lo componen. Cada brazo puede tener una pose diferente. Un vector de tiempos va indicando la duración de cada postura durante el ejercicio.
- El fichero patients.xml guarda información sobre el perfil médico del paciente, incluyendo su nombre y restricciones propias.

El componente del diseñador de terapias es el único que pertenece a una fase "offline", es decir, fuera de la propia sesión de rehabilitación. Dependiendo del modo en el que se inicie este componente, se utiliza el planificador clásico o el jerárquico y se abre una interfaz gráfica (Figura 3.2) que permite configurar la terapia. Esto planifica todas las sesiones, los ejercicios que las componen y en qué orden. Es una planificación de alto nivel. En este prototipo, ya con el plan de terapia generado, se usa la primera sesión planificada para crear un plan propio de la sesión, que se manda al módulo PELEAComp. Este trabajo incluye el diseño e implementación del planificador clásico.

PELEAComp es un componente propio de RoboComp que no se ha modificado y se encarga de la planificación de nivel medio. Es el primer componente de la fase "online" de la ejecución, ya que tiene un dominio que se encarga de planificar acciones más concretas para controlar el robot durante una sesión, como saludar, comprobar una pose, corregir una pose, levantarse, etc. Recibe del componente Ejecutivo el estado del mundo actual. Si el estado del mundo es igual al esperado, le envía la siguiente acción

previamente planificada. Si difiere, replanifica y le envía la acción siguiente nueva.

El componente Ejecutivo es otro componente RoboComp que recopila información de otros componentes con sensores como la Kinect o el robot NAO para crear un estado del mundo actual. Cuando haya terminado la acción anterior, le manda el estado del mundo a PELEA y ésta le devuelve la siguiente acción. El Ejecutivo interpreta la acción recibida y se comunica con otros componentes y el robot dependiendo de la acción recibida. Es el que envía instrucciones directas al robot para que actúe. Este componente también forma parte de este trabajo.

El robot NAO es controlado por el componente Ejecutivo. Este componente le manda instrucciones para que hable, cambie sus luces o se mueva de cierta manera. Los movimientos enviados pueden ser posturas predefinidas o un vector de ángulos que deben tener sus articulaciones. Sea cual sea el método, el robot internamente tiene un planificador de caminos para controlar los movimientos de sus articulaciones, interpolando su postura actual con la postura objetivo dependiendo de la velocidad indicada. Este es el nivel más bajo de planificación.

Normalmente, en las arquitecturas robóticas, el componente Ejecutivo se conecta con otro que es una interfaz que traduce las órdenes y comportamientos al robot. En NAOTherapist, se hace directamente en el componente Ejecutivo debido a la cantidad de dependencias que hay con las articulaciones del robot, especialmente en la comparación de posturas. Como trabajo futuro, se pretende que la arquitectura sea independiente del NAO (no así del sensor porque en THERAPIST también seguirá siendo una Kinect) pero esa generalización está fuera del marco del prototipo inicial.

El componente WinKinectComp es una interfaz que utiliza los drivers oficiales de Microsoft para analizar la imagen 3D y 2D de la cámara Kinect. Devuelve un conjunto de vectores 3D que forman el esqueleto, así como una malla de puntos con la cara detectada de la persona, si se le pide. Este componente forma parte del *framework* de RoboComp y no ha sido modificado. El componente de visión se conecta a WinKinectComp para comparar la postura del robot y la del paciente. Está planeado que

incluya también el módulo de atención del paciente con el reconocimiento de expresiones faciales y de lenguaje corporal. Este trabajo incluye el prototipo de un sistema de reconocimiento de expresiones faciales mediante aprendizaje relacional.

Existen otros dos componentes dentro de la arquitectura que aún no se han implementado en el primer prototipo de NAOTherapist. El componente de diálogo será un sistema para ofrecer una conversación coherente con el paciente, en lugar de simples mensajes pregrabados. El Ejecutivo le pedirá una frase este módulo cada vez que el robot deba hablar y este módulo hará uso de toda la información disponible (en especial del módulo de atención) para conversar de la forma más natural y adecuada posible. El componente de aprendizaje es el otro que falta por desarrollar y será el utilizado cuando el terapeuta enseñe nuevos ejercicios al sistema. Aunque aún está en fase de diseño, se espera que incluya funciones de aprendizaje más complejas pensadas para THERAPIST, como aprender qué acciones son más exitosas según la situación, peculiaridades de cada paciente, etc.

La Tabla 3.1 presenta la distribución de trabajo que ha tenido cada miembro del equipo de desarrollo de NAOTherapist. Los porcentajes indican el grado de implicación en cada parte. También se indica en qué sección de la memoria se explica la implementación de cada módulo. Una G indica que pertenece a esta memoria y una P que pertenece a la de José Carlos Pulido.

Las siguientes secciones explican detalladamente la implementación de cada una de las partes de este trabajo.

	González	Pulido	Secciones
Diseño arquitectura	50%	50%	G 3 / P 3
Base de conocimiento	50%	50%	G 3 / P 3
Diseñador de terapias			
Interfaz gráfica	30%	70%	G 3.1.1.1 / P 3.1.3
Definición del modelo	50%	50%	G 3.1.2 / P 3.1.1
Clásico	100%	0%	G 3.1
HTN	0%	100%	P 3.1.2
PELEA			
Dominio	0%	100%	P 3.2.2
Generador problema	0%	100%	P 3.2.1
Ejecutivo			
Integración PELEAComp	100%	0%	G 3.2
Integración NAO	100%	0%	G 3.2
Integración con visión	100%	0%	G 3.2.1
Visión			
Verificación de poses	20%	80%	P 3.3
Aprend. de posturas	0%	100%	P 3.3
Aprend. de exp. faciales	100%	0%	G 3.3

Tabla 3.1: Distribución de trabajo de cada componente de la arquitectura NAOTherapist.

3.1. Planificador de terapias clásico

Este módulo utiliza la Planificación Automática clásica [Ghallab et al. 2004] (una técnica de inteligencia artificial) para diseñar las terapias de rehabilitación. El sistema que se ha desarrollado para esta sección se publicó en 2014 para un paper del *workshop Knowledge Engineering for Planning and Scheduling* (KEPS) de ICAPS [Pulido et al. 2014] en el que se comparan las características de este modelo de planificación con otra aproximación basada en planificación jerárquica. NAOTherapist permite elegir uno de estos dos métodos para planificar las terapias, aunque este trabajo se centra en la Planificación Automática clásica.

A modo de vistazo preliminar, estas son las características principales que progresivamente se irán explicando a lo largo de esta sección:

- Basado en Planificación Automática clásica.
- Utiliza la base de conocimiento común de toda la arquitectura.
- Planifica qué ejercicios deben ejecutarse en cada sesión de la terapia.
- Selecciona ejercicios que sean adecuados según los criterios del médico.
- Respeta las restricciones del perfil del paciente.
- Puede proponer que el terapeuta añada nuevos ejercicios con ciertas características que le ayuden a planificar más rápido.

El componente está programado en PDDL y Java. Su salida es un plan de ejercicios para cada sesión. Estos planes se interpretan y se convierten a otro problema PDDL para que el componente PELEAComp pueda planificar las acciones del robot en una sesión de terapia con el robot NAO. Por supuesto, el planificador también puede usarse por los propios terapeutas para supervisar ellos mismos la rehabilitación.

3.1.1. Funcionamiento básico

Para facilitar la comprensión, a continuación se resume el funcionamiento básico del diseñador automático de terapias.

3.1.1.1. Configuración

En el dominio, las acciones que se manejarán serán similares a: “Ejecutar ejercicio e10”, “Iniciar fase de calentamiento”, o incluso “Aprender nuevo ejercicio corto y muy adecuado para coordinación bimanual”.

La definición del problema contiene un estado inicial a partir del cual se irán ejecutando las acciones y las metas, que definen una serie de condiciones que debe cumplir el estado final al que se quiere llegar. En este caso, este estado inicial está compuesto

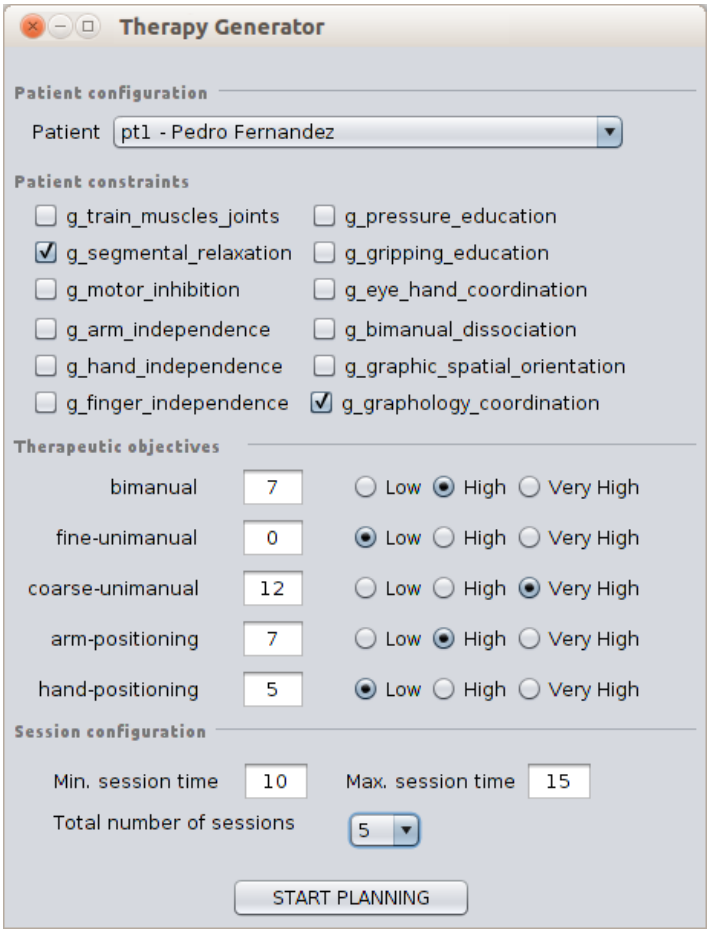

Figura 3.2: Interfaz de configuración de la terapia.

por todos los ejercicios de la base de conocimiento. Esto implica que es necesario transcodificar los ejercicios de XML a PDDL. Para ello se ha programado una aplicación en Java con interfaz de usuario que realiza la conversión y además permite seleccionar al paciente de la base de conocimiento, indicar en qué medida se quiere entrenar cada objetivo terapéutico, restricciones particulares, etc. Dicha interfaz ya se ha mostrado en la Figura 3.2. Estas características las introduce el médico y van codificadas en la sección de metas del problema en PDDL. Una vez generados los dos ficheros PDDL, se ejecuta el planificador.

3.1.1.2. Planificación

En la interfaz anterior, el médico indica cuánto debe entrenarse cada objetivo terapéutico de forma numérica. Un mayor valor indica que ese objetivo tendrá que entrenarse más en cada sesión. Estos valores (uno por cada objetivo) los utilizará el planificador automático para determinar si los ejercicios de una sesión se adecúan a la configuración establecida por el médico o no. Por simplicidad, a cada uno de estos valores se le denominará TOCL (*Therapeutic Objective Cumulative Level*).

La sesión es adecuada si se alcanzan todos los TOCLs. Para ello, cada ejercicio tiene un valor de adecuación por cada objetivo terapéutico que va de 0 (nada adecuado) a 3 (muy adecuado), inspirado en el utilizado en la escala GAS. Una sesión de rehabilitación tendrá una serie de ejercicios planificados. Para saber si una sesión es adecuada, basta con sumar separadamente todas las adecuaciones de cada objetivo terapéutico. Si el sumatorio de adecuaciones de un objetivo terapéutico supera o iguala su TOCL correspondiente, ese objetivo se habrá entrenado suficientemente. Si todos los sumatorios superan o igualan todos los TOCLs definidos por el médico, la sesión será adecuada.

En la Tabla 3.2 se muestran cuatro ejercicios, sus adecuaciones para cada objetivo terapéutico y los TOCLs que debe superar. Este conjunto de ejercicios (e5, e6, e2 y e1) sería válido para todos los TOCLs, ya que la suma de sus adecuaciones iguala o supera a todos. Sin embargo, aparte de alcanzar los TOCLs, la sesión debe cumplir con otras muchas restricciones para que sea aceptada, como que los ejercicios deben empezar siendo de calentamiento, después de entrenamiento y finalmente de enfriamiento, sesiones variables entre sí, duraciones dentro de un rango, restricciones propias del paciente, etc. Si una sesión cumple con todos estos requisitos, se aceptará como válida.

Si la base de conocimiento no dispone de suficiente número de ejercicios, el planificador puede que tarde en encontrar un plan (o quizá no exista). Por ello el dominio desarrollado contempla un mecanismo que automáticamente sugiere el "aprendizaje" de un nuevo ejercicio con características concretas. De este modo, con el uso del plani-

	A	B	C	D	E
e5	1	0	2	1	0
e6	1	0	3	0	0
e2	0	0	2	3	1
e1	0	0	0	3	0
Suma	2	0	7	7	1
TOCL	0	0	7	4	0

Tabla 3.2: Suma de adecuaciones de cuatro ejercicios y comparación con los TOCLs establecidos por el médico.

ficador, la base se irá llenando de ejercicios útiles, de forma que cada vez será menos necesario el uso de esta función y al final siempre se utilizarán ejercicios de la base de conocimiento.

3.1.2. Definición del modelo

A continuación se muestra cómo se ha modelado la terapia a alto nivel:

Metas a alcanzar por el planificador:

- Número total de sesiones.
- Duración mínima y máxima de cada sesión.
- Umbrales de TOCL a alcanzar.

Cada ejercicio se guarda en la base de conocimiento con ciertas características establecidas por el terapeuta.

Características del ejercicio:

- Duración (en minutos).

- Nivel de adecuación para cada objetivo terapéutico.
- El valor de intensidad está asociado al ritmo cardíaco medio cuando se realiza el ejercicio.
- La dificultad de un cierto paciente para realizar el ejercicio. Esta variable puede ser actualizada después de cada sesión, si es necesario.
- Cada ejercicio pertenece a un grupo de ejercicios. Estos grupos están relacionados con las capacidades motoras que el paciente necesita para poder realizar el ejercicio.
- Fases de una sesión en las que puede aparecer el ejercicio: calentamiento, entrenamiento o enfriamiento.
- Movimientos del ejercicio. Equivale a un identificador unívoco para distinguir ejercicios que podrían compartir las otras características.

Las siguientes restricciones garantizan el cumplimiento de todos los requisitos médicos y de la variabilidad de las sesiones.

Restricciones básicas:

- Cada sesión debe tener tres fases en el orden siguiente: calentamiento, entrenamiento y enfriamiento.
- La duración de cada fase y de cada sesión debe estar dentro de un rango predefinido.

Restricciones de variabilidad:

- No se permite la repetición de un cierto ejercicio en una misma sesión.
- La distribución de ejercicios debe ser variada a lo largo de las sesiones, de forma que resulten diferentes.

Restricciones del paciente:

- Impedir el entrenamiento de ciertos ejercicios, debido a que son demasiado intensos o difíciles o porque el paciente no puede hacer los de un cierto grupo.

- En una sesión, limitar la intensidad acumulada o la dificultad a un cierto umbral.

Con toda esta información, el planificador automático debe ser capaz de encontrar un plan de terapia adaptado si hay suficientes ejercicios en la base de conocimiento. En caso de que los ejercicios disponibles no sean suficientes, el planificador automático preguntará al terapeuta si necesita aprender uno nuevo con un valor sugerido para algunas características. Es a esto lo que se denomina el aprendizaje (diseño) de nuevos ejercicios.

Por ejemplo, en un plan de sesión, el planificador puede sugerir la ejecución, previo aprendizaje, de un nuevo ejercicio con un nivel de adecuación mínimo de 2 para el objetivo terapéutico "actividades bimanuales". El planificador asume que el ejercicio aprendido lo realizará el paciente y utiliza los valores estimados para planificar. Es decir, planifica con un ejercicio "abstracto", aún no aprendido, y usa características sugeridas como si fueran reales. Cuando el terapeuta guarde el nuevo ejercicio en la base de conocimiento, ésta puede tener niveles de adecuación más altos para los objetivos terapéuticos, garantizando que el plan continuará siendo válido. En sesiones futuras, los ejercicios previamente aprendidos pueden ser reutilizados, minimizando la necesidad del uso de acciones de aprendizaje y ayudando a los terapeutas a completar la base de conocimiento con un conjunto de ejercicios útiles.

Por último, después de la sesión, el terapeuta puede actualizar los valores de dificultad de los ejercicios para un paciente si es necesario. El médico puede también modificar las metas con los resultados de la evaluación de la escala GAS [Turner-Stokes 2009]. Estas actualizaciones pueden causar una replanificación de las sesiones posteriores, si la terapia previamente planificada ya no es válida.

3.1.3. Implementación

La implementación del modelo previamente expuesto es la que forma el dominio de planificación. Éste se basa fundamentalmente en variables numéricas (*fluents* en

el ámbito de la Planificación Automática) y costes de acción, introducidos en PDDL 2.1 [Fox et al. 2003]. Es cierto que estos requisitos aún tienen una falta de soporte en muchos de los planificadores actuales, pero en este dominio particular son especialmente útiles para operar directamente con los valores cuantitativos de los objetivos terapéuticos. El planificador CBP [Fuentetaja et al. 2010] soporta estas características y su método de búsqueda está guiado por una selección de acciones extraídas del grafo de un plan relajado (esto es, el boceto de un plan preliminar en el que no tienen por qué cumplirse todas las restricciones). Las variables numéricas son útiles para controlar la longitud de la sesión, añadir restricciones de variabilidad y establecer una preferencia dinámica para ciertas acciones.

No existen muchos dominios que exploten las variables numéricas y ciertas características (como la función de sugerir nuevos ejercicios al vuelo, el manejo de una gran cantidad de variables numéricas, etc.) hacen de él un dominio innovador, de interés en el área de la Planificación Automática. La implementación del modelo ha permitido explorar las capacidades del planificador automático que se ha utilizado, por lo que la solución final ha estado muy condicionada por los tiempos de planificación.

En general, el criterio de diseño más importante que se ha seguido es que cada sesión individual debe cumplir siempre los objetivos terapéuticos y la duración de sus fases tiene que estar dentro del margen indicado. El criterio secundario ha sido el de forzar la variabilidad entre las sesiones de terapia para evitar la monotonía y mejorar la efectividad del tratamiento. El mecanismo de sugerencia de nuevos ejercicios se explica más detalladamente en una sección posterior.

Para clarificar la siguiente explicación, se muestra un plan para una única sesión en la Figura 3.3. La terapia completa contendrá todos los planes de todas las sesiones planificadas, respetando todas las dependencias entre ellas:

Cada línea corresponde a una acción. Estas acciones, a veces tienen ciertos atributos como el identificador del ejercicio (E0, E11, etc.). La acción de aprendizaje tiene unos atributos diferentes: el objetivo terapéutico (O) para el que el nuevo ejercicio

```
 0: (SESSION-START)
 1: (WARMUP-PHASE)
 2: (WARMUP-DATABASE-EXERCISE E0)
 3: (TRAINING-PHASE)
 4: (TRAINING-DATABASE-EXERCISE E11)
 5: (TRAINING-DATABASE-EXERCISE E12)
 6: (TRAINING-DATABASE-EXERCISE E10)
 7: (TRAINING-DATABASE-EXERCISE E9)
 8: (LEARN-TRAINING-EXERCISE O_SPATIAL_HAND A_MEDIUM
      D_LONG)
 9: (COOLDOWN-PHASE)
10: (COOLDOWN-DATABASE-EXERCISE E15)
11: (SESSION-END)
```

Figura 3.3: Plan de alto nivel para una sesión de rehabilitación.

debe ser adecuado, el nivel de adecuación (A) para ese objetivo terapéutico y su duración (D). Esta acción de aprendizaje puede elegir entre 5 objetivos terapéuticos, 3 adecuaciones y 3 duraciones diferentes. El valor de cada adecuación y cada duración está asignado en el problema PDDL. Las subsecciones siguientes explican detalles más precisos sobre la implementación del problema y del dominio.

3.1.3.1. Definición del problema

El fichero de problema contiene las metas que deben cumplirse en el estado final y el estado inicial a partir del que se empieza a planificar.

Metas: El médico decide el número total de sesiones y la duración mínima y máxima de cada fase. Estos datos son almacenados en el estado inicial del problema en PDDL. También debe escoger restricciones dependiendo del perfil del paciente y, sobre todo, determinar la cantidad de entrenamiento para cada objetivo terapéutico en cada sesión. Esto último se tiene en cuenta en las metas del problema PDDL.

Hay un *fluent* para cada TOCL que va acumulando todos los correspondientes valores de adecuación de los ejercicios planificados en una sesión. Un TOCL se alcan-

zará si la suma de adecuaciones del objetivo terapéutico correspondiente lo iguala o supera. Por ello basta con asignar una meta con un umbral inferior para cada TOCL. Como ejemplo, estas podrían ser las metas para una sesión de 30 minutos:

```
(>= (TOCL o_bimanual) 15)
(>= (TOCL o_unimanual_fine) 7)
(>= (TOCL o_spatial_arm) 7)
```

Para clarificar, la primera meta indica que la variable numérica de tipo TOCL llamada o_bimanual debe tener un valor mayor o igual a 15 para que la sesión sea válida. Como se puede ver, las metas numéricas en PDDL permiten una gran flexibilidad para configurar el rango de valores deseados de cada TOCL. También sería muy sencillo establecer un límite superior para cada objetivo (con una condición de menor o igual) o incluso evitar el entrenamiento de un cierto objetivo (igualándolo a cero), pero esto tiene menos sentido médico ya que estos valores son solo una forma de representar la prioridad del entrenamiento de los objetivos terapéuticos y no necesitan estar directamente relacionados con la intensidad de los ejercicios. Para esto último ya existe una variable numérica diferente.

Estado inicial: contiene todos los ejercicios almacenados en el sistema. Estos ejercicios están en la base de conocimiento XML, por lo que para este trabajo ha sido necesario desarrollar también un transcodificador de XML a PDDL que genere el problema correctamente. La base de conocimiento está completamente controlada por el terapeuta, que añade ejercicios cuando el sistema se lo sugiere o cuando él lo encuentre conveniente. Contiene las características de los ejercicios mencionadas anteriormente. El valor de dificultad de cada ejercicio va de 0 a 100 y se guarda en el perfil del paciente, pero para simplificar consideramos que son leídos en el problema PDDL antes de la tarea de planificación. Para asegurar el cumplimiento de las restricciones de variabilidad, hay dos variables numéricas adicionales que representan el número de sesión y la posición del ejercicio en la última sesión donde apareció. Cada ejercicio tiene un predicado que restringe su aparición a las fases de calentamiento, entrenamiento o enfriamiento. Un ejercicio de calentamiento puede no estar aconsejado para la fase

de enfriamiento, por lo que en este trabajo se han diferenciado completamente ambas fases, en lugar de clasificar ejercicios de poca intensidad como de calentamiento y enfriamiento indistintamente.

El sistema asume que la información de la base de conocimiento es coherente, por lo que el terapeuta debe asegurarse de que los ejercicios son correctos cuando los añada. Por ejemplo, los ejercicios de calentamiento no deberían ser demasiado intensos. Con estas consideraciones, el plan de la sesión empezará con una intensidad suave, seguido de una fase de entrenamiento más intensa y terminando con ejercicios suaves de nuevo. A continuación se muestra el ejemplo de un ejercicio terapéutico genérico (e7) modelado en PDDL:

```
(e_phase e7 p_training)
(e_group e7 g_arm_independence)
(= (e_last_session e7) 2)
(= (e_last_position e7) 4)
(= (e_intensity e7) 48)
(= (e_difficulty e7) 39)
(= (e_duration e7) 4)
(= (e_adequacy e7 o_bimanual) 0)
(= (e_adequacy e7 o_unimanual_fine) 3)
(= (e_adequacy e7 o_unimanual_coarse) 0)
(= (e_adequacy e7 o_spatial_arm) 1)
(= (e_adequacy e7 o_spatial_hand) 0)
```

Este fragmento de código PDDL indica las diferentes características de este ejercicio. Por ejemplo, la primera línea es un predicado con dos atributos que indica que el ejercicio e7 está en la fase de entrenamiento y la segunda el grupo al que pertenece. Las demás asignan valores numéricos a los *fluents*. El *fluent* de las adecuaciones incluye dos objetos (el ejercicio y el objetivo terapéutico de la adecuación) y después el valor asignado. El estado del mundo también contiene diversas variables de control sobre la sesión actual como la duración mínima y máxima de cada fase.

3.1.3.2. Definición del dominio

Acciones: En primer lugar se explican las acciones normales, con las que se puede planificar una terapia cuando hay ejercicios suficientes en la base de conocimiento. Las acciones de aprendizaje se explican al final.

Todas las acciones están fuertemente basadas en *fluents*, dado que hay precondiciones numéricas y costes de acción. Existen dos grupos básicos de acciones: para controlar el flujo de la sesión y para añadir ejercicios desde la base de conocimiento.

Las acciones de control de flujo permiten el cambio de fase dependiendo de unas precondiciones. Estas acciones son:

- session-start
- warmup-phase
- training-phase
- cooldown-phase
- session-end

Como ejemplo se muestra la acción cooldown-phase. Se puede ejecutar cuando la variable time_training es mayor que el tiempo mínimo de entrenamiento (determinado por el terapeuta en el fichero de problema). Las otras acciones funcionan de forma similar:

```
(:action cooldown-phase
    :parameters ()
    :precondition (and
        (phase_training)
        (>= (time_training) (min_time_training))
    )
    :effect (and
        (phase_cooldown)
        (not (phase_training))
    )
)
```

Básicamente, esta acción establece que puede ejecutarse si se está en la fase de

entrenamiento y el tiempo de entrenamiento ha superado el mínimo. El efecto provoca que se inicia la fase de enfriamiento y que se deje de estar en la de entrenamiento.

La forma básica de añadir ejercicios a una sesión es a través de acciones que los seleccionan de la base de conocimiento. Solo pueden seleccionarse ejercicios para la fase actual (calentamiento, entrenamiento o enfriamiento), por eso se han separado las acciones según la fase. Estas son las acciones de inserción de ejercicios desde la base de conocimiento:

- warmup-database-exercise
- training-database-exercise-a
- training-database-exercise-b
- cooldown-database-exercise

Como ejemplo, a continuación se muestra la acción training-database-exercise-a:

```
(:action training-database-exercise-a
    :parameters (?e - e_training)
    :precondition (and
        (phase_training)
        (< (time_training) (max_time_training))
        (> (- (index_session) (e_last_session ?e) ) 2)
        (< (index_exercise) (e_last_position ?e))
    )
    :effect (and
        (assign (e_last_session ?e) (index_session))
        (assign (e_last_position ?e) (index_exercise))

        (increase (time_training) (e_duration ?e))
        (increase (index_exercise) 1)

        (increase (TOCL o_bimanual)
          (e_adequacy ?e o_bimanual))
        (increase (TOCL o_unimanual_coarse)
          (e_adequacy ?e o_unimanual_coarse))
        (increase (TOCL o_unimanual_fine)
          (e_adequacy ?e o_unimanual_fine))
        (increase (TOCL o_spatial_arm)
          (e_adequacy ?e o_spatial_arm))
        (increase (TOCL o_spatial_hand)
```

```
            (e_adequacy ?e o_spatial_hand))
        )
    )
```

Las precondiciones de esta acción indican que se puede ejecutar si se está en la fase de entrenamiento y el tiempo que se lleva entrenado es menor que el tiempo máximo de esta fase. Adicionalmente hay dos precondiciones más que sirven para asegurar la variabilidad de las sesiones:

- El ejercicio no puede haber sido usado en las tres últimas sesiones.
- En la fase de entrenamiento, un ejercicio no puede ser entrenado en la misma posición que en la última en la que apareció. Para calentamiento y enfriamiento esta condición no es aplicable porque un ejercicio largo puede alcanzar por si mismo el tiempo mínimo de la fase y no podría cambiar el orden (por ejemplo, en la fase de calentamiento, ese ejercicio aparecerá siempre en la primera posición).

Normalmente la última precondición se modelaría con una precondición negativa (posición del ejercicio no igual a la última), sin embargo muchos planificadores no son compatibles con esta funcionalidad, por lo que la acción se ha dividido en dos (una para cuando el ejercicio aparece antes que en la vez anterior y otra después). Internamente, los planificadores que soportan precondiciones negativas hacen esto mismo, por lo que no hay ningún problema de rendimiento en esta decisión de diseño.

Los efectos de esta acción asignan la última sesión y la última posición al ejercicio, incrementan el tiempo de entrenamiento lo que dure el ejercicio e incrementan el sumatorio de las adecuaciones.

Acciones de aprendizaje: Ayudan al terapeuta a añadir nuevos ejercicios útiles a la base de conocimiento mientras el sistema se va utilizando. Permiten continuar con la planificación si no encuentra un plan válido. También son tres acciones, según sea la fase en la que debe aparecer el ejercicio:

- learn-warmup-exercise

- learn-training-exercise
- learn-cooldown-exercise

Es preferible usar ejercicios en la base de conocimiento en lugar de aprender nuevos, pero no es necesario explorar todo el espacio de búsqueda antes de probar una acción de aprendizaje. Esto se ha controlado utilizando un coste de acción más alto para ellas. El planificador intenta minimizar el coste total del plan, por lo que poco a poco las acciones de aprendizaje irán siendo menos utilizadas. A continuación se muestra un ejemplo de acción de aprendizaje:

```
(:action learn-warmup-exercise
    :parameters (?o - objective ?t - l_duration ?a - l_adequacy)
    :precondition (and
        (learning_activated)
        (phase_warmup)
        (< (time_warmup) 6)
        (< (learnings_current) (learnings_max))
    )
    :effect (and
        (increase (learnings_current) 1)

        (increase (time_warmup) (l_duration_value ?t))
        (increase (TOCL ?o) (l_adequacy_value ?a))
        (increase (index_exercise) 1)

        (increase (total-cost)
          (- (+ (l_adequacy_value ?a) 10) (l_duration_value ?t)))
    )
)
```

La acción es parecida a las de inserción desde la base de conocimiento, aunque incrementa solamente los valores de los parámetros que haya seleccionado el planificador automático. Como ya se ha explicado, existen 5 objetos para objetivos terapéuticos, 3 adecuaciones y 3 duraciones que tienen valores asignados en el problema PDDL. El planificador selecciona estos objetos según su heurística y planifica la acción con ellos.

Para incrementar la variabilidad, el coste de acción será más alto cuando el nuevo ejercicio permita alcanzar las metas del problema más rápido. De este modo, los ejer-

cicios más largos y menos adecuados para el objetivo principal son preferidos. Sin este coste de acción dinámico, el planificador podría escoger aprender siempre el ejercicio más corto y el más adecuado, en detrimento de la variabilidad.

Tanto en el sistema real, como en NAOTherapist, es conveniente tener una fase de preprocesado para aprender todos los ejercicios necesarios antes de empezar cada sesión y así únicamente ejecutar los ejercicios en las posiciones planificadas. El plan podría empezar con todas las acciones de aprendizaje al principio y después los ejercicios planificados para la sesión. Sin embargo, esta es una tarea de planificación realmente complicada porque primero necesitaría planificar la creación de los nuevos ejercicios antes de que apareciese la necesidad de la creación de los mismos.

3.1.4. Estrategia de planificación

La planificación clásica tiene que enfrentarse con un problema bastante complejo en este dominio. Planificar solo una sesión es algo relativamente fácil, pero una terapia real está compuesta por unas 20 sesiones. La primera aproximación fue planificar una terapia completa, generando planes que contenían más de 250 acciones. Planificar múltiples sesiones de una vez causa un incremento no lineal del tiempo de planificación porque existen muchas dependencias entre ellas. Los planificadores actuales suelen ir planificando las acciones desde la primera hasta la última, de modo que cuando no encuentran solución vuelven atrás (hacen *backtracking*) un cierto número de pasos y prueban otras acciones.

En este dominio, el planificador tiene que volver atrás si los ejercicios seleccionados para una sesión no son válidos. El problema aparece cuando el planificador vuelve atrás más lejos de lo necesario, quizá varias sesiones válidas, forzando a replanificar de nuevo esas sesiones para encontrar una alternativa válida para una sesión posterior. Un *backtracking* más pequeño de solo unas pocas acciones podría resolver la situación permitiendo reordenar los ejercicios, seleccionando otros o planificando el aprendizaje de uno nuevo para continuar.

Adicionalmente al problema de *backtracking*, hay otro inconveniente que es necesario resolver: los nuevos ejercicios aprendidos no pueden ser reutilizados en sesiones posteriores porque la creación de nuevos objetos en PDDL aún es un problema abierto. Una forma de solucionar esto podría ser definir algunos objetos de tipo "hueco de aprendizaje". El planificador podría asignar nuevos predicados a estos huecos para controlar la variabilidad y los atributos del ejercicio para reutilizar estos ejercicios aprendidos en sesiones posteriores. Pero esto produce una distinción conceptual entre un ejercicio de la base de conocimiento y un ejercicio aprendido. Estos ejercicios, una vez aprendidos, son semánticamente idénticos, por lo que no tiene sentido realzar esta distinción en el modelo. Además, con este método el número de ejercicios que pueden ser aprendidos está limitado por el número de huecos reservados en el problema. Por estas razones, este método fue descartado.

Finalmente se ha optado por una estrategia divide y venderás (Figura 3.4) para planificar cada sesión individualmente, pero teniendo en cuenta todas las dependencias existentes entre ellas. En esta estrategia, se llama al planificador una vez por cada sesión que se quiera planificar. Cada vez que el planificador devuelve un plan, éste se interpreta para determinar qué ejercicios de la base de conocimiento y qué ejercicios aprendidos se han planificado. En la siguiente sesión, se genera un nuevo fichero de problema para actualizar los predicados y funciones de los ejercicios en la última acción planificada, y añade los nuevos ejercicios a la base de conocimiento. Después, el planificador se ejecuta de nuevo con el fichero de problema para la siguiente sesión. Por tanto, para cada sesión, se genera un problema en PDDL con los nuevos ejercicios y actualizaciones de la base de conocimiento. Los experimentos muestran que esta estrategia es mucho más rápida que planificar todas las sesiones de una vez, sin afectar, en la práctica, a la calidad de los planes generados. La capacidad de aprender nuevos ejercicios da a las sesiones algunas propiedades de localidad que pueden ser explotadas para evitar el *backtracking* entre sesiones que consume tanto tiempo de planificación.

Es cierto que planificando la terapia completa se podría obtener un plan de mayor

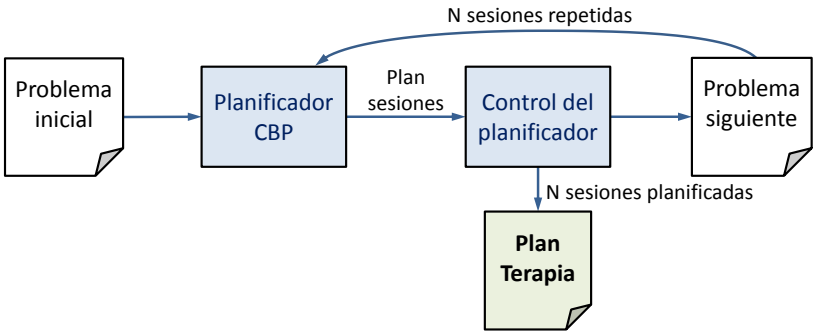

Figura 3.4: Estrategia de planificación divide y vencerás.

calidad (por ejemplo, encontrando un plan que solo utilizase ejercicios de la base de conocimiento), pero el objetivo en este dominio es encontrar un plan válido para una sesión, no uno cercano al óptimo. Además, los experimentos muestran que si la base de conocimiento es suficientemente grande, las costosas acciones de aprendizaje dejan de planificarse. El planificador debe encontrar un plan rápido para que sea útil para el terapeuta, de otro modo el equipo médico seguiría utilizando su forma tradicional, manual o improvisada, de diseñar sus terapias.

3.2. Componente Ejecutivo

Este componente se ha programado en Python dado que es muy sencillo comunicarse directamente con el NAO de esta manera. RoboComp aísla cada componente y permite que se comunique con los demás, estén escritos en el lenguaje de programación que sea, por lo que usar Python solo trae ventajas.

En esencia, el Ejecutivo es el que lleva todo el peso del control de una sesión. Para ello, se comunica con PELEA y con el componente de visión. El Ejecutivo no toma ninguna decisión acerca de qué debe o no hacer el robot, esa tarea recae sobre PELEA, por eso, cuando el sistema ya ha terminado de realizar la última acción, este componente le pide la acción siguiente a PELEA. Para ello el Ejecutivo tiene que tener

una representación más o menos fiel de cómo es el estado del mundo y enviársela a PELEA para que planifique las acciones posibles. Una vez obtenida la acción, le manda las instrucciones correspondientes al robot para que las ejecute.

3.2.1. Estado del mundo

La recreación del estado del mundo requiere extraer datos de los sensores, haciendo uso del componente de visión y su sistema de atención paciente (aún no implementado) y del robot para los predicados exógenos. Estos predicados no cambian en los efectos del dominio de planificación de sesiones de PELEA, sino que aparecen externamente, siendo el componente Ejecutivo el encargado de añadirlos. En este trabajo no se ha programado el dominio de PELEA, pero el Ejecutivo debe tener un conocimiento exhaustivo de este dominio, dado que debe generar estados del mundo compatibles para PELEA y a interpretar todas las acciones de este dominio. PELEA recibe estados del mundo en formato PDDL. Estos son los predicados exógenos:

- detected_patient: Lo determina el componente de visión.
- identified_patient: Lo determina el componente de visión.
- patient_distracted: Recogido a través del módulo de atención del componente visión.
- emergency_situation: Cuando el paciente se marcha o el módulo de atención determina que sucede algo grave.
- posture_changed: El paciente se ha levantado o se ha sentado sin que se lo dijese el robot.
- paused_session: Se pausa la sesión. Este predicado aparece en los efectos del dominio, pero su eliminación del estado del mundo corre a cargo del componente Ejecutivo. Esto se produce cuando el terapeuta reanuda la sesión (presiona una tecla del teclado).
- uncontrolled_situation: Cuando la sesión se ha pausado, el terapeuta puede cancelar la sesión presionando una tecla del teclado. En este caso aparece este predicado

que obliga a PELEA a finalizar la sesión.

- posture_state: Indica si el paciente o el robot están levantados o sentados. El del paciente se determina con la visión. El del robot se le pregunta al NAO directamente mediante el módulo NaoQi ALRobotPosture.
- correct_pose: Lo determina el componente de visión.

Los únicos predicados exógenos que aún no se cambian en el prototipo desarrollado son aquellos que dependen del módulo de atención del componente de visión. Los demás sí se actualizan dependiendo de los sensores o según indiquen los efectos de cada acción. Es decir, para generar el estado del mundo, primero se determina la acción que recibe el Ejecutivo de PELEA, después éste aplica todos los efectos (no exógenos) que contiene la acción para actualizar el estado del mundo. Los predicados no exógenos que pueden detectarse mediante algún sensor se revisan para comprobar que la acción se ha ejecutado correctamente. Una vez actualizados todos los no exógenos, se revisan los sensores para determinar el estado de los predicados exógenos.

3.2.2. Acciones

Cada acción que recibe de PELEA tiene asociado un comportamiento diferente del robot y del sistema en general. Algunas de las acciones son simplemente de control, pero otras requieren el uso de los sensores, movimientos, hablar, etc. A continuación se muestra una lista con todas ellas y la forma que tiene el componente Ejecutivo de tratarlas. Gracias a esta descripción, se puede entender en qué consiste el dominio de PELEA, aunque en este trabajo no se describe su implementación en detalle porque no forma parte del mismo.

- detect-patient: Pide el estado del atributo detected_patient al módulo de visión.
- identify-patient: Pide los datos de usuario al módulo de visión. Esta parte no está implementada en el primer prototipo, ya que su flujo de ejecución es diferente. En concreto, el paciente se configura al inicio cuando se diseña la terapia. El

Ejecutivo lee los datos del paciente de la base de conocimiento en XML.

- greet-patient: Saluda con voz y el brazo al paciente. Le llama por su nombre.
- start-training: Es una acción de control. Anuncia verbalmente al paciente que se va a empezar la sesión.
- introduce-exercise: Antes de empezar un ejercicio, el robot le presenta un poco en qué consiste y para qué sirve. El Ejecutivo carga el texto que debe leer de la base de conocimiento de ejercicios y lo reproduce.
- stand-up: Utiliza el módulo NaoQi AlRobotPosture para levantarse. El uso de este módulo permite que el NAO se levante desde cualquier postura, ya esté tumbado, sentado, boca abajo, etc.
- sit-down: Igual que la acción stand-up, pero se sienta.
- start-exercise: Es una acción de control. Se reinician los contadores de poses y temporizadores para el ejercicio que se va a realizar.
- execute-pose: Esta acción es de las más importantes. El Ejecutivo manda al robot la postura que debe adoptar con ambos brazos mediante el módulo NaoQi AlMotion y la función angleInterpolationWithSpeed. El robot es el que se encarga de planificar la interpolación de los movimientos de bajo nivel. Cada postura se mantiene el tiempo que venga determinado en la base de conocimiento para ese ejercicio. Iterativamente se va comprobando con el módulo de visión si el paciente tiene la misma pose o no todo el rato que dure esa pose. Si el paciente ha mantenido la pose el tiempo necesario, al terminar se añade el predicado correct_pose al estado del mundo. Si no se está manteniendo, no se añade este predicado.
- correct-pose: Se ejecuta cuando la pose anterior no se ha realizado correctamente o no se ha mantenido el tiempo necesario. Al comparar la pose, el módulo de visión entrega al componente Ejecutivo una serie de números que indican cuánto se ha desviado cada articulación del ángulo esperado. Atendiendo a estos números, el componente Ejecutivo manda un mensaje al robot para que éste le diga al paciente unas correcciones básicas sobre cómo modificar su postura “estira más el brazo izquierdo”, “levanta más el brazo”, etc. Vuelve a hacer la misma funcionalidad

que execute-pose, aunque si el paciente falla demasiado el módulo Ejecutivo puede pasar al siguiente ejercicio o, en caso extremo, pausar la sesión para que venga el terapeuta.

- finish-pose: Es una acción de control. Reinicia los predicados y variables temporales de la pose anterior.
- finish-exercise: El robot comunica al paciente que han terminado el ejercicio.
- finish-training: El robot simula que se seca el sudor de la frente y le dice que el entrenamiento ha terminado.
- say-good-bye: El robot se despide del paciente verbalmente y con gestos.
- finish-session: El robot se sienta para esperar al próximo paciente.
- claim-stand-up: Si el paciente está sentado y el ejercicio requiere que esté levantado, el robot le pide al paciente que se levante.
- claim-sit-down: Si el paciente está levantado y el ejercicio requiere que esté sentado, el robot le pide al paciente que se siente.
- claim-attention: Si el módulo de atención del componente de visión determina que el paciente está distraído, el robot llama su atención.
- pause-session: La sesión se pausa, por lo que el terapeuta debe venir para comprobar qué ha pasado. El sistema se queda a la espera de que el terapeuta reanude o cancele la sesión mediante el teclado.
- resume-session: Se elimina el predicado pause_sesion para reanudarla después de que el terapeuta dé el visto bueno.
- cancel-session: Se añade el predicado cancel_session para cancelar la sesión si el terapeuta considera que no puede continuarse con ella.

Como ya se ha comentado, toda la arquitectura se pone en marcha mediante un *script* de *shell*. El prototipo desarrollado termina su ejecución cuando se termina la sesión.

3.3. Reconocedor de expresiones faciales

Se ha desarrollado el prototipo de un sistema capaz de aprender y clasificar automáticamente nueve expresiones y posturas faciales mediante minería de datos relacional. Con una cámara Kinect se han capturado imágenes de la cara, de las que luego se han extraído los puntos más característicos usando las librerías del SDK Face Tracking de Microsoft. Tomando estos puntos como base, se ha modelado una base de conocimiento con predicados relacionales para el algoritmo de aprendizaje relacional Tilde. Tras programar una aplicación que rellene esta base de conocimiento con multitud de ejemplos de entrenamiento, Tilde extrae de ellos un árbol relacional capaz de clasificar nueve expresiones faciales diferentes.

3.3.1. Visión general

El proceso de clasificación consta de cuatro fases diferenciadas (Figura 3.5):

1. Obtención de puntos característicos: La cámara Kinect ofrece un vídeo en tiempo real y las librerías del Face Tracker lo procesan para devolver una malla de puntos característicos de la cara en tiempo real.

2. Obtención de predicados para Tilde: El usuario indica el fotograma que quiere convertir en ejemplo de entrenamiento y el extractor de predicados obtiene los predicados para Tilde de los puntos característicos. Como es aprendizaje supervisado, el usuario también indica de qué clase es el ejemplo que está capturando.

3. Entrenamiento del modelo con ACE-Tilde: Los ejemplos de entrenamiento forman la base de conocimiento que se le envía a Tilde. El algoritmo entrenará un árbol de decisión relacional.

4. Validación del modelo entrenado: Con el árbol extraído se puede clasificar automáticamente cualquier expresión facial entrenada.

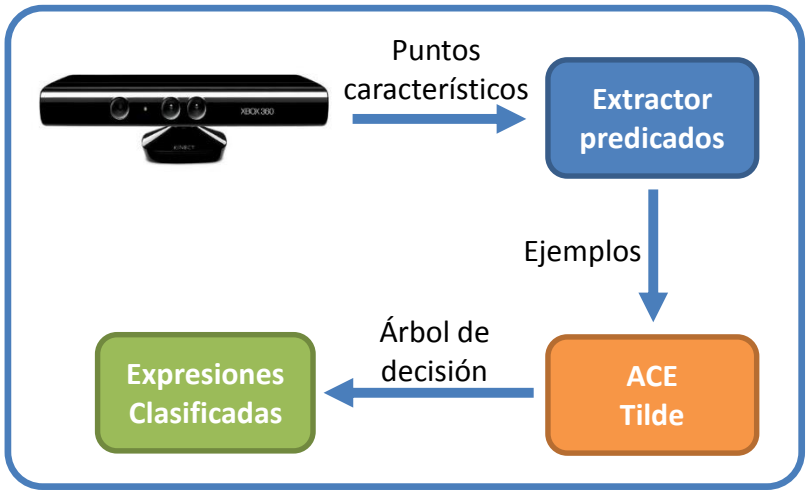

Figura 3.5: Diseño del sistema de reconocimiento de expresiones faciales.

3.3.2. Modelo relacional

Esta subsección explica cómo se ha ido desarrollando el modelado de la base de conocimiento relacional hasta llegar a la versión definitiva. Para ello se detallan tres apartados: la selección de puntos característicos, las expresiones faciales elegidas para clasificar y los predicados que se le envían a Tilde para que aprenda.

3.3.2.1. Puntos característicos

Lo primero que se hizo fue experimentar con las distintas mallas de puntos descritas en la sección 2.3.2 para utilizar solo una de las dos o una combinación de ambas. Se optó por utilizar la Malla de Modelo ya que el propio SDK tenía una selección de puntos con nombres propios de ella, pero no de la otra. Inicialmente se probó a clasificar con los 121 puntos de la malla, pero resultó evidente que eran demasiados objetos (puntos) sobre los que establecer relaciones en el árbol. La selección de 70 puntos proporcionada por el SDK seguía siendo muy grande, así que se optó por utilizar el mínimo número de puntos posible y luego ir aumentándolo a medida que fueran siendo necesarios. De este modo se terminó con una selección de tan solo 9 puntos característicos, como

puede verse en la Figura 3.7 y la Figura 3.9, que se realizó ad-hoc tras una evaluación empírica reducida:

p0 Centro superior frente

p16 Centro ceja izquierda

p49 Centro ceja derecha

p23 Interior ojo izq.

p56 Interior ojo der.

p7 Centro labio superior

p31 Izquierda boca

p64 Derecha boca

p9 Punto superior barbilla

3.3.2.2. Expresiones a clasificar

Se han seleccionado un total de 10 expresiones faciales entre las que se distinguen tres grupos (Figura 3.6): las que no miran de frente, las que miran de frente con las cejas bajadas y las que miran de frente con las cejas subidas. Es evidente que, de forma natural, se trata de un problema multietiqueta-multiclase. Sin embargo para probar si funcionaba la idea del modelo relacional, estas diez expresiones se han tomado como clases únicas y diferentes. Esta selección ha estado condicionada por las limitaciones de la malla que proporcionaba el SDK Face Tracker, ya que, por ejemplo, los puntos no lograban ajustarse bien a la boca cuando expresaba tristeza (comisuras hacia abajo).

Inicialmente se empezó a clasificar solo la dirección a la que miraba la persona: izquierda, arriba, abajo y derecha ($\leftarrow$, $\uparrow$, $\downarrow$, $\rightarrow$). Los buenos resultados hicieron que se añadiese otro grupo de expresiones para cuando se mira de frente: neutral, sonriente y

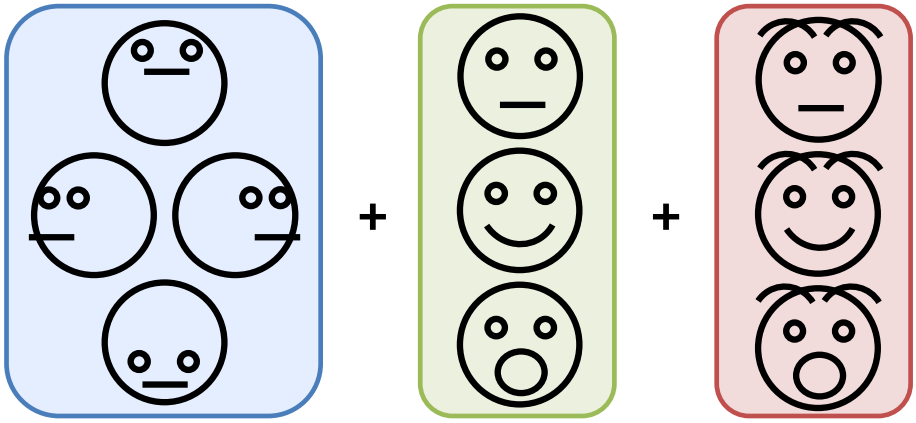

Figura 3.6: Las diez expresiones faciales a clasificar.

con la boca abierta (:|, :), :O). Como Tilde también clasificaba muy bien, se optó por utilizar otro grupo más de expresiones muy parecidas a las anteriores para dificultarle el trabajo: las tres anteriores con las cejas levantadas (<:|, <:), <:O).

3.3.2.3. Predicados para ACE-Tilde

Con los nueve puntos característicos seleccionados y las expresiones a clasificar, lo que queda por definir son los predicados que permitan a Tilde establecer relaciones entre los puntos.

La principal dificultad radica en que Tilde internamente utiliza un razonamiento simbólico, es decir, no numérico. Las coordenadas X e Y de cada punto son numéricas, así como muchas de las medidas que se pueden hacer con ellas, por lo que será necesario discretizar estas características para Tilde. Hay dos formas de hacer esto: una es dejando que Tilde sea el que realice esta discretización y la otra es hacerla manualmente a la hora de construir los predicados. Se decidió hacerla manualmente para entender mejor cómo funcionaba Tilde y tener más control para saber si los predicados que se estaban modelando realmente eran útiles o no.

Al igual que con los puntos característicos, en todo momento se ha intentado

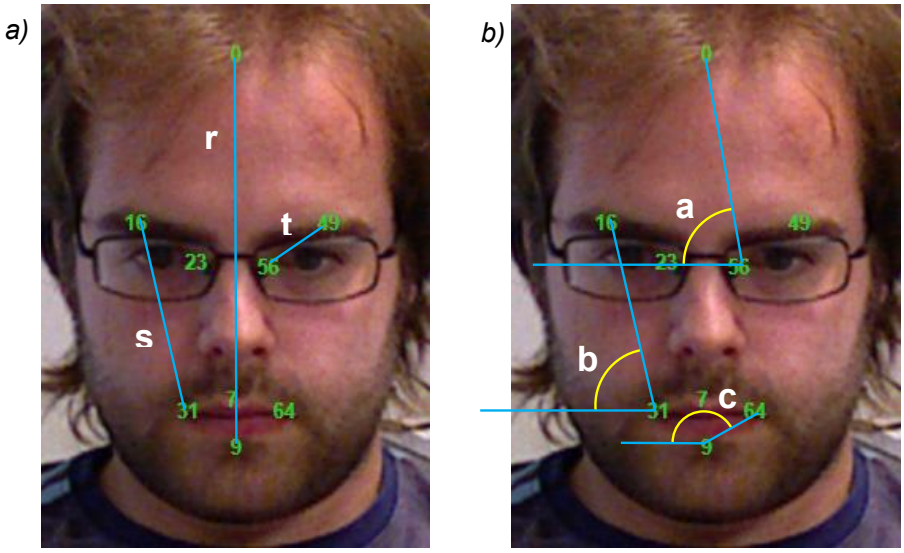

Figura 3.7: a) Ejemplos de distancias. b) Ejemplos de ángulos.

utilizar la menor cantidad de predicados y procurar que fuesen lo más genéricos posibles, es decir, que hubiera poco conocimiento del dominio. La idea era obtener un conjunto de puntos con relaciones entre ellos poco evidentes a simple vista, para ver si Tilde era capaz de sacar las conclusiones oportunas. Con todo esto en mente se programaron dos tipos de predicados: distancias y ángulos.

- **Distancias**: Para cada par de puntos se calcula la distancia entre ellos y se normaliza dividiendo entre la distancia más grande (ejemplos en la Figura 3.7a). En ese ejemplo, s y t estarían divididos entre r, ya que de las 36 distancias posibles r es la más grande. Se han discretizado en diez pasos de tamaño 0,1.

- **Ángulos**: Para cada par de puntos se calcula el ángulo de la recta que los une con el eje horizontal (ejemplos en la Figura 3.7b). Cada uno de los 36 ángulos se divide entre π para normalizarlos. Se han discretizado en diez pasos de tamaño 0,1.

En la Figura 3.8 se puede ver un ejemplo de cómo son los predicados (la imagen está espejada). Por ejemplo, `angle5_6(p49,p64)` significa que el ángulo normalizado

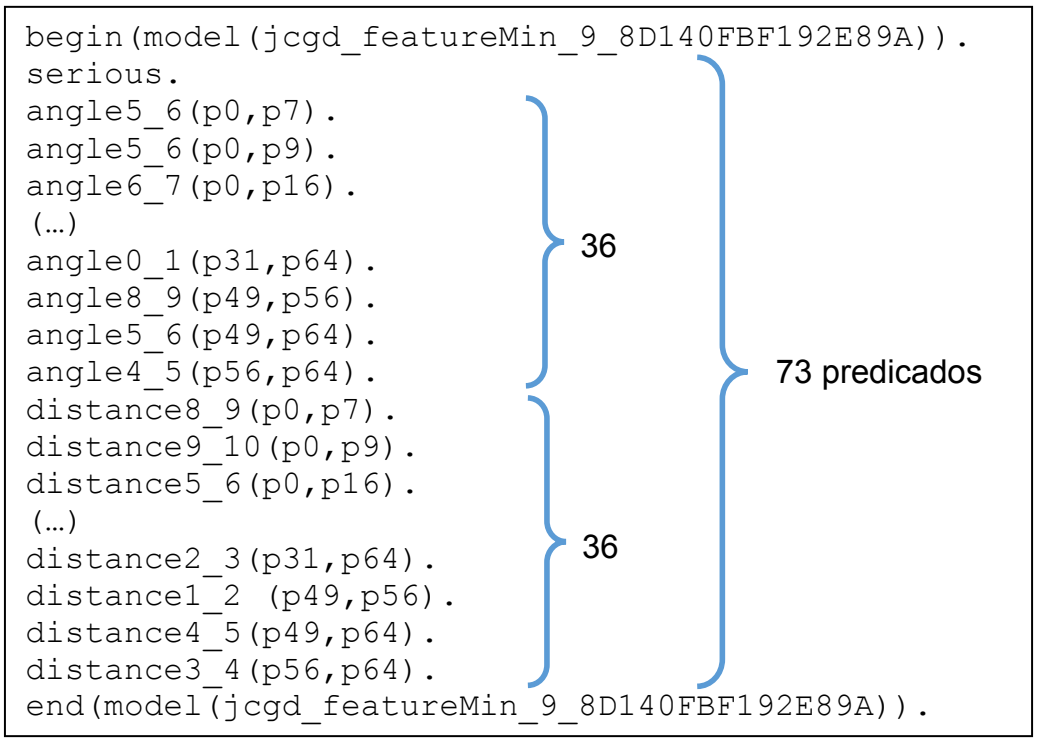

Figura 3.8: Predicados de un ejemplo de entrenamiento.

de la recta que pasa por la ceja derecha y la derecha de la boca está entre 0,5 y 0,6. Asimismo, el predicado `distance9_10(p0,p9)` significa que la distancia normalizada entre la frente y el labio inferior está entre 0,9 y 1,0. En este caso, discretizar en 10 rangos ha funcionado bien, ya que las diferencias entre pares de puntos eran muy grandes o muy pequeñas, y a menudo estables entre expresiones.

En total, cada ejemplo tiene 36 predicados de distancias, 36 de ángulos y 1 que define de qué clase es el ejemplo, sumando un total de 73 predicados. ACE-Tilde sí tiene en cuenta el orden de los objetos en los predicados, por lo que podría pensarse que hace falta el doble de predicados para añadirlos también con los puntos al revés. Esto no es necesario por la forma en la que se generan los ejemplos, ya que el punto con el índice más bajo siempre está a la izquierda del más alto. Puesto que nunca aprende nada en orden distinto, en principio no hay necesidad de tener en cuenta el orden inverso de los

puntos. Además, tal como se verá en los experimentos, puede resultar útil.

Por último, mencionar que se programó un tercer tipo de predicado que indicaba si un punto estaba a la izquierda o encima de otro. Los experimentos mostraron que este predicado era inútil, así que se optó por dejarlo de lado.

3.3.3. Extractor de predicados

La obtención de las imágenes, extracción de puntos característicos y cálculo de predicados para Tilde se ha programado en una aplicación en C# para tomar ejemplos de forma rápida (Figura 3.9).

La interfaz incluye varios elementos para seleccionar la malla de puntos deseada. La imagen se puede tomar en HD o normal y controlar el zoom y el ángulo vertical de la Kinect con su motor. La parte más importante es donde están los botones y contadores para tomar nuevos ejemplos. Cada vez que se pulsa uno de estos botones se toma la imagen del vídeo actual, se generan los predicados para Tilde y se adjunta el nuevo ejemplo a un fichero .kb que representa la base de conocimiento. Cada botón corresponde a una pose, por lo que para tomar los ejemplos de entrenamiento basta con poner la cara deseada y pulsar el botón con el emoticono adecuado.

En este trabajo la adquisición de ejemplos es muy sencilla ya que la Kinect devuelve un vídeo y cada fotograma es un potencial ejemplo de entrenamiento. Naturalmente, se podría calcular un ejemplo por cada fotograma del vídeo, pero hacerlo manualmente permite generar una base de conocimiento con menos ruido, ya que lo que interesaba constatar en este trabajo era cómo de bien aprendía Tilde con los predicados que se estaban modelando.

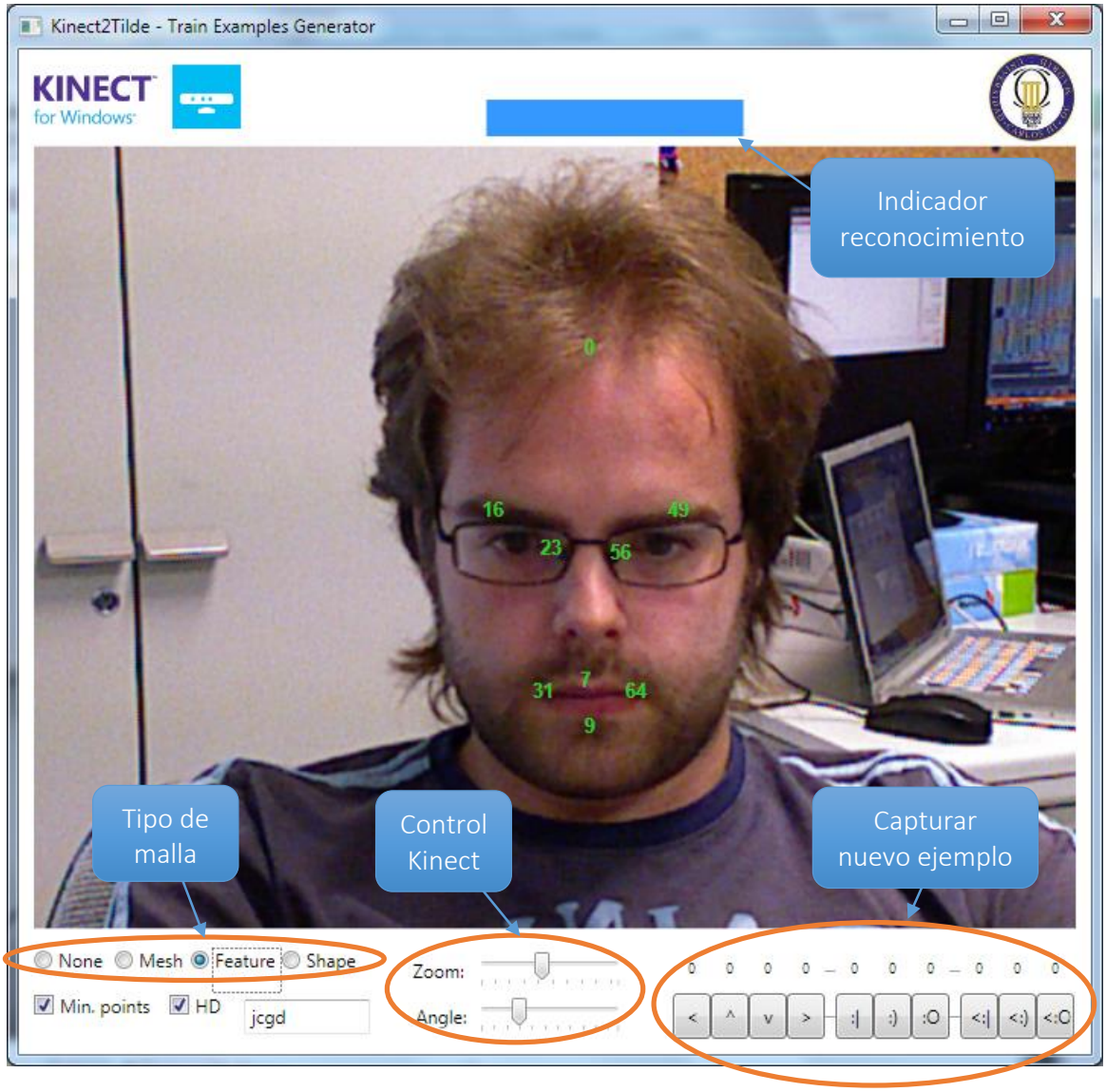

Figura 3.9: Puntos característicos e interfaz del extractor de predicados.

Capítulo 4

Evaluación del sistema

En este capítulo se presenta la evaluación del proyecto. Para seguir con la estructura general de la memoria, los experimentos se han ordenado según las distintas partes del trabajo y la prueba final de concepto del prototipo NAOTherapist, la cual también demuestra el correcto funcionamiento del componente Ejecutivo.

4.1. Diseñador Terapias

En los experimentos de esta sección se prueba el funcionamiento del dominio con varios planificadores automáticos, prestando especial atención a la variabilidad de las sesiones, a los efectos de las acciones de aprendizaje sobre el plan final y a las condiciones que afectan a los tiempos de planificación.

Como ya se ha explicado en el capítulo Arquitectura, planificar todas las sesiones de una vez con CBP o Metric-FF no dio resultados satisfactorios para más de 5 o 6 sesiones (no terminaba de planificar). También se intentó modelar el dominio para que planificase primero todas las acciones de aprendizaje y al final los ejercicios, sin embargo esto provoca mucha confusión en las heurísticas de los planificadores, ya que era necesario planificar primero una "generación de objetos" (los nuevos ejercicios)

antes de que se detecte que éstos son necesarios.

En este punto conviene recordar que las acciones de aprendizaje de ejercicios permiten que, aunque no haya ejercicios suficientes en la base de conocimiento, el planificador siempre consiga encontrar un plan. Para evitar que esta funcionalidad fuese sobreexplotada, se aplica un coste de acción más alto a las acciones de aprendizaje. De ese modo el planificador debería preferir utilizar los ejercicios de la base de conocimiento, aunque le cueste más trabajo encontrar una combinación adecuada.

La Figura 4.1 demuestra la eficacia de los costes de acción en el plan generado. Se ha planificado un plan para el mismo problema (misma sesión) usando Metric-FF [Hoffmann 2003] y CBP [Fuentetaja et al. 2010]. Metric-FF no utiliza costes de acción, mientras que CBP sí los tiene en cuenta. Como se puede ver, los planes son muy diferentes entre sí. Metric-FF utiliza indistintamente ejercicios de aprendizaje y de la base de conocimiento, por lo que el plan está muy poco optimizado. Por otro lado, las heurísticas de CBP permiten reducir el número total de acciones de aprendizaje de forma que en este caso solo sea necesaria una. Además, los atributos sugeridos en el ejercicio también están optimizados para favorecer la variabilidad: su duración es larga (d_long) en lugar de corta o media y adecuación es media (a_medium) en lugar de máxima. Como ya se explicó, lo más eficiente sería que todos los nuevos ejercicios fuesen cortos y muy adecuados, pero las sugerencias siempre serían las mismas. CBP, por tanto, es el planificador que se ha escogido para que planifique las terapias.

El tiempo que CBP tarda en generar una terapia es muy dependiente de los ejercicios que haya en la base de conocimiento y, dada la cantidad de relaciones existentes entre ellos, es difícil encontrar una correlación precisa que sea indicativa del tiempo que tardará en planificar. Normalmente cada sesión tarda menos de un segundo, sin embargo, se han observado ciertas condiciones que aumentan el tiempo de planificación.

Una de ellas es la cantidad de TOCL a alcanzar. Si estos valores son muy altos tendrá menos variedad de ejercicios para seleccionar y, puesto que en este dominio las restricciones de variabilidad son estrictas, la cantidad de aprendizajes necesarios

Metric-FF

```
 0: SESSION-START
 1: WARMUP-PHASE
 2: LEARN-WARMUP-EXERCISE
    O_SPATIAL_HAND A_MAXIMUM D_LONG
 3: TRAINING-PHASE
 4: TRAINING-DATABASE-EXERCISE E7
 5: LEARN-TRAINING-EXERCISE
          O_UNIMANUAL_COARSE
          A_MAXIMUM D_MEDIUM
 6: LEARN-TRAINING-EXERCISE
          O_UNIMANUAL_COARSE
          A_MAXIMUM D_SHORT
 7: TRAINING-DATABASE-EXERCISE E12
 8: LEARN-TRAINING-EXERCISE
          O_UNIMANUAL_FINE
          A_MAXIMUM D_MEDIUM
 9: COOLDOWN-PHASE
10: LEARN-COOLDOWN-EXERCISE
          A_SPATIAL_HAND
          A_MODERATE D_LONG
11: SESSION-END
```

CBP IPC-2011

```
 0: (SESSION-START)
 1: (WARMUP-PHASE)
 2: (WARMUP-DATABASE-EXERCISE E0)
 3: (TRAINING-PHASE)
 4: (TRAINING-DATABASE-EXERCISE E11)
 5: (TRAINING-DATABASE-EXERCISE E12)
 6: (TRAINING-DATABASE-EXERCISE E10)
 7: (TRAINING-DATABASE-EXERCISE E9)
 8: (LEARN-TRAINING-EXERCISE
          O_SPATIAL_HAND
          A_MEDIUM D_LONG)
 9: (COOLDOWN-PHASE)
10: (COOLDOWN-DATABASE-EXERCISE E15)
11: (SESSION-END)
```

Figura 4.1: Comparación de planificadores para un mismo problema. Uno en Metric-FF, sin costes de acción, y otro en CBP.

también aumentará. De hecho, los TOCLs tienen un umbral a partir del cual sería imposible encontrar un plan, ya que la adecuación máxima de un ejercicio es 3, y solo caben un número limitado de ejercicios en una sesión aunque estos sean muy cortos. En terapias reales, sin embargo, los TOCLs se mantienen dentro de un intervalo planificable, por lo que no supone un problema.

Otro factor que aumenta el tiempo de planificación es la cantidad de ejercicios que haya en la base de conocimiento. Endurecer las restricciones del paciente limita los ejercicios disponibles y en general dificulta la planificación pero, sin tenerlas en cuenta para la experimentación, se han observado tres situaciones diferentes:

- Cuando existen pocos ejercicios en la base de conocimiento (unos 20 en total), en las sesiones 3, 4 y 5 se produce el mayor consumo de tiempo de planificación. En estas sesiones las restricciones de variabilidad impiden reutilizar ejercicios que se usaron hace poco, por lo que el planificador tiene que planificar acciones de

aprendizaje. Una vez aprendidos ejercicios útiles, éstos se reutilizan más tarde y se resiente el tiempo de planificación. Los ejercicios aún disponibles en la base de conocimiento no tienen por qué ser útiles para las metas propuestas, por lo que el planificador tiene que explorar estos ejercicios para finalmente seleccionar una acción de aprendizaje. La relación de tiempos de planificación varía muchísimo con cada problema, aunque el caso más extremo que se ha encontrado no superó los 10 minutos.

- La segunda situación se da cuando la base de conocimientos está casi vacía (solo 3 ejercicios). Paradójicamente el tiempo de planificación medio de toda la terapia es mucho menor que en la situación anterior. La diferencia es que cuando quedan ejercicios disponibles cuyas adecuaciones no ayudan a alcanzar las metas, éstos se planifican en un plan intermedio y el planificador solo se da cuenta de que no son útiles cuando ya no puede alcanzar las metas, obligándole a hacer *backtracking*. Cuando la base de conocimiento está vacía, se va rellenando con ejercicios útiles para ese problema concreto desde el principio, de manera que su reutilización en sesiones posteriores es válida y no requiere *backtracking* por insertar ejercicios poco adecuados.
- Cuando existen suficientes ejercicios en la base de conocimiento (unos 50), la planificación es rápida porque siempre se encuentra algún ejercicio disponible y adecuado para la terapia en cuestión. Se han hecho pruebas con más de 200 ejercicios en la base de conocimiento con características aleatorias y se ha determinado que aumentar el número de ejercicios no afecta negativamente al tiempo de planificación

La preferencia por usar ejercicios de la base de conocimiento en una única sesión se puede ver en la Figura 4.1, pero este experimento no permite determinar si los nuevos ejercicios aprendidos son útiles para sesiones posteriores. En la Tabla 4.1 se ve claramente que la idea funciona. En esta terapia se empieza con pocos ejercicios en la base de conocimiento. Una “e” o una “L” con un número representa un ejercicio

Ejercicios →

Sesiones ↓

	1	2	3	4	5	6	7	8
1	e0	e9	e11	e12	e10	e7	e15	
2	e4	e2	e5	e6	**L**	**L**	**L**	e13
3	e1	e3	e8	**L**	**L**	**L**	**L**	e16
4	**L**	**L**	**L**	**L**	**L**	**L**	e17	
5	e0	e11	e12	e10	e9	**L**	e15	
6	e4	e2	e6	L19	e7	e5	L20	e13
7	e1	e3	L24	e8	L23	L22	e16	
8	L25	L26	L30	L27	L28	L29	e17	
9	e0	e12	e10	L31	e11	e9	e15	
10	e4	e2	L19	L20	e6	e7	e13	
11	e1	e3	L22	L23	L24	e8	e16	
12	**L**	L25	L26	L29	L30	L27	L28	e17
13	e0	e10	L21	e12	e9	e11	e15	
14	e4	e2	e7	e6	L20	L19	e13	
15	e1	e3	L24	e8	L22	L23	e16	
16	L25	L27	L31	L26	e5	L29	L30	e17
17	e0	e11	e12	L28	e10	e9	e15	
18	L32	e4	e2	L19	e6	L20	e7	e13
19	e1	e3	L22	L23	L24	e8	e16	
20	L25	L26	e5	L29	L30	e17		

Tabla 4.1: Plan de terapia con pocos ejercicios en la base de conocimiento: 5 de calentamiento, 10 de entrenamiento y 5 de enfriamiento.

almacenado en la base de conocimiento (inicial o aprendido, respectivamente). Una "L" sola representa el aprendizaje y ejecución de un nuevo ejercicio. Cada sesión implica un nuevo problema y una nueva ejecución del planificador debido a la estrategia divide y vencerás que se ha seguido.

En la sesión 1, el planificador solo utiliza ejercicios de la base de conocimiento porque éstos son útiles para alcanzar los TOCLs. En las sesiones 2 y 3, necesita aprender debido a que no quedan ejercicios disponibles debido a las restricciones de variabilidad. En la sesión 4 casi todos los ejercicios se han usado en las últimas tres sesiones, por lo que necesita seguir aprendiendo nuevos. En la sesión 5, el planificador ya puede usar un grupo de ejercicios de la sesión 1 de nuevo, pero variando el orden de la fase de entrenamiento porque los ejercicios no pueden aparecer en la misma posición que

Sesiones \ Ejercicios	1	2	3	4	5	6	7	8	9	10
1	e3	e18	e33	e22	e23	e28	e31	e49		
2	e8	e32	e19	e27	e35	e17	e47			
3	e9	e25	e34	e15	e16	e44				
4	e7	e29	e21	e24	e26	e48	e42			
5	e0	e1	e3	e20	e30	e33	e18	e23	e28	e49
6	e4	e8	e32	e19	e17	e31	e47			
7	e5	e13	e16	e34	e25	e22	e36	e44		
8	e7	e2	e24	e26	e21	e15	e27	e42		
9	e1	e3	e30	e33	e18	e20	e23	e28	e49	
10	e0	e8	e19	e32	e29	e17	e47			
11	e5	e13	e34	e16	e22	e36	e25	e44		
12	e4	e7	e26	e27	e31	e24	e42			
13	e1	e3	e33	e30	e20	e23	e28	e18	e49	
14	e0	e8	e32	e17	e15	e19	e47			
15	e5	e13	e21	e34	e16	e25	e22	e44		
16	e4	e7	e24	e26	e27	e31	e42			
17	e1	e3	e30	e33	e28	e18	e23	e49		
18	e0	e8	e19	e32	e17	e15	e35	e47		
19	e5	e13	e34	e21	e25	e22	e16	e44		
20	e4	e7	e26	e24	e31	e27	e42			

Tabla 4.2: Plan de terapia con muchos ejercicios en la base de conocimiento: 15 de calentamiento, 22 de entrenamiento y 14 de enfriamiento.

en la última vez. En las siguientes sesiones, los ejercicios aprendidos anteriormente se empiezan a reutilizar porque los TOCL no han cambiado y siguen siendo adecuados para ellos. A partir de esta sesión, no son necesarias más acciones de aprendizaje. Nótese que las sesiones planificadas son muy diferentes entre ellas.

Los nuevos aprendizajes en las sesiones 5 y 12 muestran que las acciones de aprendizaje no están completamente prohibidas, por lo que no hace falta explorar todas las combinaciones en la base de conocimiento antes de utilizar una acción de aprendizaje. Esta restricción tan relajada mejora muchísimo los tiempos de planificación, aunque tiene un pequeño sobrecoste en la calidad (coste total) del plan generado. Por supuesto, las funciones de aprendizaje pueden inhabilitarse automáticamente después de un tiempo para forzar a buscar solo en la base de conocimiento pero eso es algo que no se

Ejercicios →

Sesiones ↓

	1	2	3	4	5	6	7	8	9
1	e0	e1	**L**	**L**	**L**	**L**	e2	**L**	
2	**L**	**L**	**L**	**L**	**L**	**L**	**L**	**L**	
3	**L**	**L**	**L**	**L**	**L**	**L**	**L**	**L**	
4	**L**	**L**	**L**	**L**	**L**	**L**	**L**	**L**	
5	e0	L5	e1	L6	L4	L3	L7		
6	L8	L9	L10	L13	L11	L14	L12	L15	
7	L16	L17	L18	L21	L19	L22	L20	L23	
8	L24	L25	L26	L29	L27	L30	L28	L31	
9	e0	L6	L4	L5	L3	e1	L7		
10	**L**	L8	L9	L11	L14	L13	L10	L15	
11	**L**	L16	L17	L19	L22	L12	L21	L20	L23
12	**L**	L24	L25	L27	L30	L29	L26	L31	
13	e0	e1	L6	L18	L28	L4	L7		
14	L8	L9	L13	L10	L11	L14	L3	L15	
15	L33	L16	L22	L21	**L**	L19	L12	L23	
16	L17	L29	L30	**L**	L26	L27	**L**	e2	L31
17	L32	L24	L28	L6	**L**	L18	e1	L4	L7
18	L8	e0	L20	L13	**L**	L10	L15		
19	L33	L16	L35	L22	L14	L3	L23		
20	L9	L37	L29	L30	L5	L27	e2	L31	

Tabla 4.3: Plan de terapia con muy pocos ejercicios: 1 de calentamiento, 1 de entrenamiento y 1 de enfriamiento.

ha probado para el desarrollo de este prototipo.

Adicionalmente, se utiliza solo el primer plan que encuentra el planificador CBP. Este planificador puede mejorar los planes iterativamente si tiene tiempo, pero experimentalmente se muestra que el primer plan devuelto ya es suficientemente bueno para ver cómo funciona el dominio modelado.

El comportamiento de las acciones de aprendizaje con muchos ejercicios en la base de conocimiento se puede ver en la Tabla 4.2 y con muy pocos en la Tabla 4.3. Efectivamente, cuando tiene muchos no necesita hacer ninguna acción de aprendizaje. Cuando está casi vacía, se observa un comportamiento similar a de la Tabla 4.1, aprendiendo al inicio y reutilizando estos ejercicios después, aunque se producen más aprendizajes esporádicos.

4.2. Componente Ejecutivo

El componente Ejecutivo es un componente software cuya evaluación ha consistido en probar una tras otra todas las acciones para que realice su tarea de forma correcta. Su implementación ha sido imprescindible para poder probar la arquitectura completa y algunas partes ajenas a este trabajo, como el dominio de PELEA.

Normalmente, para la simulación de eventos exógenos se suele utilizar el software MDPSim[1] y el lenguaje PPDDL (*Probabilistic PDDL*) [Younes et al. 2004], que incluye probabilidades en los efectos. Esto provoca que las acciones ya no sean deterministas, igual que en el mundo real. Sin embargo, MDPSim es demasiado antiguo para utilizarlo con *fluents* y poder probar el software. Dado que el Ejecutivo debe mantener un estado interno del mundo, se ha implementado también una función de simulación ad-hoc para el dominio de PELEA que sustituye a MDPSim. El componente Ejecutivo añade o no un evento exógeno dependiendo de una cierta probabilidad. Las pruebas de funcionamiento de la arquitectura se han acelerado mucho gracias a la función de simulación programada en este componente.

Al final de este capítulo se presenta una descripción general de la prueba de concepto de la arquitectura que demuestra el correcto funcionamiento de este componente.

4.3. Reconocedor Expresiones

A continuación se muestran los resultados de los experimentos preliminares que se han llevado a cabo para el reconocedor de expresiones faciales con la cámara Kinect. Se han hecho con 2 personas (un varón de 27 años y una mujer de 15 años) tomando 1000 ejemplos a cada una. Cada expresión facial tenía 100 ejemplos por cada persona. La base de conocimiento, por tanto, tiene 2000 ejemplos en total.

[1] http://www.tempastic.org/mdpsim/ (Accedida el 2/9/2014)

4.3.1. Entrenamiento con 1612 ejemplos

En este experimento, se indica a ACE que cada ejemplo de las dos personas tendrá una probabilidad del 80 % para ser de entrenamiento y del 20 % para ser de test. Adicionalmente, para evitar árboles demasiado grandes (sobreentrenamiento), se ha forzado a que clasificase al menos 30 ejemplos por hoja.

Re\Pr	←	↑	↓	→	:|	:)	:O	<:|	<:)	<:O	
←	159										159
↑		163									163
↓	8		89		49	2			8		156
→				170							170
:|					147			10			157
:)						164			1		165
:O							153			8	161
<:|					79			63	20		162
<:)	6					30		19	97		152
<:O							7			160	167
	173	163	89	170	275	196	160	92	126	168	1612

Tabla 4.4: Matriz de confusión para entrenamiento (80 % entren.).

En la Tabla 4.4 y la Tabla 4.5 se pueden ver las matrices de confusión de los ejemplos de entrenamiento y de los de test respectivamente. El objetivo de esta prueba es evaluar la capacidad que tiene Tilde para clasificar muchos ejemplos diferentes. Como se puede ver, <:| es la única clase mal clasificada. ↓ y <:) son aceptables, aunque su precisión está por debajo del 70 %, tal y como se puede ver en la Tabla 4.6.

El clasificador ha conseguido clasificar bien 9 clases de 10 con un 81 % de precisión. Las clases <:|, ↓ y :| son demasiado parecidas, por lo que Tilde tiende a confundirlas. Concretamente tiene preferencia por :|, por eso esta clase tiene un 10 % de falsos positivos. La intensidad de la relación según el coeficiente de Cramer es bastante alta.

Algunas confusiones entre <:) y :) pueden deberse a la calidad de los ejemplos

Re\Pr	←	↑	↓	→	:|	:)	:O	<:|	<:)	<:O	
←	41										41
↑		37									37
↓	3		25		12	3					44
→				30							30
:|					40			3			43
:)						35			2		35
:O							37			2	39
<:|					24			18	10		52
<:)	1					11		6	30		48
<:O										34	34
	45	37	25	30	76	49	37	27	41	36	403

Tabla 4.5: Matriz de confusión para test (80 % entren.).

Clase	Positivos reales	Positivos falsos
←	100%	1%
↑	100%	0%
↓	57%	0%
→	100%	0%
:|	93%	10%
:)	100%	4%
:O	95%	0%
<:|	35%	3%
<:)	63%	3%
<:O	100%	1%
Precisión: 81%		
Coef. Cramer: 0,86		

Tabla 4.6: Porcentajes de precisión detallada para test (80 % entren.).

de entrenamiento. En concreto, tomar ejemplos con cejas levantadas era más difícil de lo que parecía. Mantener expresiones complejas durante un buen rato fue un poco complicado para las personas que se prestaron a hacer los experimentos. A veces no podían contener la risa cuando debían estar serios o relajaban demasiado las cejas. Al presionar rápidamente el botón de toma de ejemplos para molestar lo menos posible a los voluntarios, a veces se tomaban ejemplos con una clasificación incorrecta o con expresiones muy ambiguas. Por otro lado, la baja precisión de la clase ↓ probablemente podría solucionarse añadiendo algún punto más de la Malla de Modelo.

Destacan las clases ←, ↑, →, :), :O y <:O, que obtienen casi un 100 % de precisión. En la Figura 4.2 se puede ver el árbol relacional que genera Tilde.

Básicamente, primero busca un par de puntos A y B que estén muy cerca entre sí. Si los hay y B está una décima parte más lejos de otro punto C, entonces mira a la izquierda. Si esto último no se cumple, mira a la derecha. Esta primera asociación tiene sentido. Al mirar a la derecha o a la izquierda los puntos de la malla se amontonan, acortando sus distancias. La segunda condición parece contradictoria, ya que la cara es simétrica. Sin embargo, como se ha comentado antes, el punto con el índice más bajo se escribe siempre a la izquierda del predicado y el más alto a la derecha. En este caso esta relación de orden artificial parece resultar útil para discriminar el lado al que mira, ya que acierta el 100 % de las veces.

Si ningún par de puntos está tan junto, entonces la cara no está girada. La siguiente rama intenta clasificar <:|, aunque aparentemente es una consulta sin mucho sentido. Probablemente por eso esta clase tiene una precisión tan baja. Luego evalúa una serie de distancias para determinar si la boca está abierta y después si las cejas están levantadas o bajadas. Si no se cumplen estas condiciones, la cara está mirando arriba. Después, intenta clasificar ↓ y :| de forma poco clara (de nuevo se asocia :| con una expresión con baja precisión). Finalmente mediante unas mediciones de ángulos, determina las sonrisas con cejas levantadas y sin ellas. Es muy interesante el hecho de que Tilde considera la cara neutral :| como una especie de comodín a la acuden las clases más

```
distance0_1(-A,-B) ?
       +--yes: distance1_2(B,-C) ?
       |       +--yes: [←] 173.0
       |       +--no:  [→] 170.0
       +--no:  distance7_8(-D,-E) ?
               +--yes: distance1_2(D,-F) ?
               |       +--yes: [<:|] 53.0
               |       +--no:  distance3_4(-G,E) ?
               |               +--yes: distance2_3(E,-H) ?
               |               |       +--yes: [<:O] 168.0
               |               |       +--no:  [:O] 160.0
               |               +--no:  [↑] 163.0
              +--no:  angle9_10(-I,-J) ?
                       +--yes: distance4_5(J,-K) ?
                       |       +--yes: distance5_6(-L,J) ?
                       |       |       +--yes: angle7_8(-M,L) ?
                       |       |       |       +--yes: [<:)] 76.0
                       |       |       |       +--no:  [<:|] 39.0
                       |       |       +--no:  angle5_6(I,-N) ?
                       |       |               +--yes: [:)] 196.0
                       |       |               +--no:  [<:)] 50.0
                       |       +--no:  [↓] 89.0
                       +--no:  [:|] 275.0
```

Figura 4.2: Árbol de decisión relacional C4.5 (80 % entren.).

difíciles de clasificar.

Se observa que las relaciones aparentemente más lógicas son las que tienen mayor precisión. Las más oscuras van perdiendo efectividad, aunque como ya se ha comentado, se ha forzado a que las hojas del árbol clasificasen al menos 30 ejemplos para evitar un posible sobreentrenamiento. La precisión de la clasificación sin esta restricción es de más de un 90 %.

4.3.2. Entrenamiento con 40 ejemplos

Como experimento adicional, se ha probado Tilde con muy pocos datos de entrenamiento. Para ello se ha reducido la base de conocimiento a solo 40 ejemplos: 2 por persona y clase. La Tabla 4.7 muestra la clasificación para los ejemplos de entrenamiento.

Re\Pr	←	↑	↓	→	:|	:)	:O	<:|	<:)	<:O	
←	4										4
↑		4									4
↓			4								4
→				4							4
:|					2			2			4
:)						4					4
:O							4				4
<:|								4			4
<:)									4		4
<:O										4	4
	4	4	4	4	2	4	4	6	4	4	40

Tabla 4.7: Matriz de confusión para entrenamiento (40 ejemp. entren.).

De nuevo :| y <:| se han confundido, por lo que parece indicar que estas expresiones realmente son muy parecidas, al menos para una de las personas del experimento. El resto de las clases las clasifica perfectamente. El árbol resultante aparece en la Figura 4.3.

El árbol es similar al del experimento anterior. De nuevo, mirar hacia uno u otro lado y la boca abierta lo hace en el mismo punto por los motivos que ya se han explicado. La segunda mitad es diferente, fundamentalmente porque mezcla <:| y :| varias veces y estas son las clases que más le cuesta clasificar. De nuevo, la cara neutral sirve de comodín si fallan todas las demás consultas Prolog.

```
distance05_1(-A,-B) ?
      +--yes: distance1_2(B,-C) ?
      |       +--yes: [←] 4.0
      |       +--no:  [→] 4.0
      +--no:  distance7_8(-D,-E) ?
              +--yes: distance3_4(E,-F) ?
              |       +--yes: distance1_2(E,-G) ?
              |       |       +--yes: [:O] 4.0
              |       |       +--no:  [<:O] 4.0
              |       +--no:  distance1_2(D,-H) ?
              |               +--yes: [<:)] 2.0
              |               +--no:  [↑] 4.0
              +--no:  angle9_10(-I,-J) ?
                      +--yes: distance4_5(J,-K) ?
                      |       +--yes: distance2_3(J,-L) ?
                      |       |       +--yes: [<:|] 2.0
                      |       |       +--no:  distance6_7(-M,K) ?
                      |       |               +--yes: [<:)] 2.0
                      |       |               +--no:  [:)] 4.0
                      |       +--no:  [↓] 2.0
                      +--no:  distance6_7(-N,-O) ?
                              +--yes: distance1_2(O,-P) ?
                              |       +--yes: [↓] 2.0
                              |       +--no:  [<:|] 4.0
                              +--no:  [:|] 2.0
```

Figura 4.3: Árbol de decisión relacional C4.5 (40 ejemp. entren.).

4.4. Prototipo NAOTherapist

La prueba final de concepto de NAOTherapist evalúa el prototipo y la arquitectura general. Para ello, el autor de este trabajo ha asumido el rol de médico y terapeuta y otra persona el de paciente. El prototipo aún es demasiado primitivo para ofrecer una interacción completa con los pacientes, ya que solo se ha investigado parte del módulo

de atención del componente de la visión. Las pruebas realizadas buscan evaluar la correcta conexión de los componentes y el funcionamiento de todos ellos en conjunto.

Para la prueba de concepto, se han puesto suficientes ejercicios en la base de conocimiento para que las acciones de aprendizaje no sean necesarias, ya que el módulo de aprendizaje aún no ha sido desarrollado. La ejecución de una sesión completa de rehabilitación suele ser de unos 20 minutos. Esto es demasiado engorroso para pruebas del prototipo, por lo que para demostrar su correcto funcionamiento basta con que las sesiones duren entre 5 y 10 minutos.

Gracias a la unión de todas las partes desarrolladas, se ha logrado tener un prototipo funcional capaz de ejecutar un escenario básico consistente en:

- Configuración de la terapia y posterior generación y planificación con el diseñador de terapias (planificación clásica y jerárquica).
- Traducción del plan de la primera sesión de terapia de alto nivel a un problema para PELEA.
- Planificación de nivel medio utilizando PELEA para el control de la sesión.
- Componente Ejecutivo que mantiene un estado del mundo interno, obtiene eventos exógenos de los sensores e interpreta y ejecuta las acciones que devuelve PELEA en el robot.
- Comparación de la pose del robot y la del paciente mediante un módulo de visión.

El reconocedor de expresiones faciales que se ha desarrollado en este trabajo demuestra que el aprendizaje relacional es útil para reconocer expresiones, pero aún no ha sido integrado en la arquitectura por estar en una fase de investigación inicial. El objetivo es conseguir un módulo de atención del paciente que incluya también un reconocedor de posturas corporales (de lenguaje corporal).

Tras las pruebas queda claro que el robot es capaz de hacer una sesión de rehabilitación completa y de forma fluida. A pesar del enorme trabajo de diseño e implementación de cada componente y lo dispar de sus características, se han logrado interconectar de forma exitosa todos los componentes, por lo que el resultado cumple

Figura 4.4: Demostración del funcionamiento del prototipo.

con las expectativas de este prototipo. En el capítulo Discusión se comentan diversos aspectos a mejorar de esta arquitectura y el trabajo futuro de NAOTherapist.

Se puede acceder a unos vídeos con demostraciones del prototipo funcional de NAOTherapist en https://www.youtube.com/user/NAOTherapist/.

Capítulo 5

Discusión

A continuación se exponen las conclusiones de cada parte del trabajo, así como del proyecto NAOTherapist en general. También se comentan las líneas de investigación futuras y las mejoras que pueden aplicarse al prototipo.

5.1. Planificador de Terapias

En resumen, se ha conseguido modelar en PDDL un dominio para planificar terapias de rehabilitación que además contempla en la planificación el aprendizaje de nuevos ejercicios. Para ello se ha necesitado recurrir a una estrategia de divide y vencerás que planifique las sesiones de una en una, evitando replanificar aquellas que ya hayan sido validadas.

Como trabajo futuro, es posible ir mejorando las soluciones poco a poco, en lugar de quedarse siempre con la primera que devuelva el planificador. Por ejemplo, estableciendo un tiempo máximo para planificar cada sesión, sería posible ir encontrando varios planes cada vez mejores que redujesen el coste total, y con ello el número de acciones de aprendizaje.

Esta técnica permitiría también asignar un coste de acción dinámico a las acciones

de inserción de ejercicio de la base de datos. Así, los ejercicios más intensos serían menos costosos cuanto más cerca de la mitad de la sesión estuviesen y más costosos cuando estuviesen al principio y al final. Esto no se implementó porque primero había que determinar experimentalmente si el coste era capaz de influir en el número de acciones de aprendizaje en el primer plan devuelto.

También, el uso de un planificador con *landmarks* podría evitar la planificación con divide y vencerás. Los *landmarks*, en planificación, son estados que se detectan en una fase de preprocesado a los que forzosamente se debe llegar en algún momento para alcanzar las metas. En este dominio, las acciones de cambio de fase tienen que ejecutarse secuencialmente: inicio-calentamiento-entrenamiento-enfriamiento-fin. Cada una de ellas es un *landmark* que podría ayudar muchísimo a planificar sin divide y vencerás (toda la terapia de golpe). Desgraciadamente, los *landmarks* no se han desarrollado para variables numéricas, y estas acciones tienen precondiciones numéricas, por lo que una interesante línea de investigación paralela podría ser el desarrollo de *landmarks* numéricos para dominios como este.

También sería interesante realizar pruebas con el planificador automático OPTIC [Benton et al. 2012], ya que dispone de heurísticas específicas para *fluents* que podrían reducir el tiempo de planificación.

5.2. Componente Ejecutivo

El desarrollo de este componente ha sido especialmente complicado debido a la enorme cantidad de interacciones que tenía con componentes ajenos a este trabajo. El Ejecutivo es el mayor exponente de la arquitectura RoboComp ya que cada uno de los componentes con los que interactúa puede estar en un sitio diferente (PELEA en máquina Linux, la visión en máquina Windows y NaoQi en el propio robot NAO). Determinar el uso más adecuado de RoboComp, así como del sistema de control del robot NAO, ha tenido una gran dificultad técnica. Sin embargo, el componente Ejecutivo es

capaz de conectarse a todos los módulos y controlar la sesión de terapia correctamente.

El funcionamiento del Ejecutivo es muy dependiente de las capacidades de los componentes a los que se conecta, por lo que su trabajo futuro depende mucho de qué nuevas funciones se hagan en otros módulos. Sin embargo, el principal refinamiento que puede añadírsele es que no se conecte al NAO de forma directa, sino que lo haga a través de un componente que sirva de interfaz para el propio NAO. De este modo, simplemente sustituyendo este nuevo componente se podría usar otro robot sin hacer más cambios. Esto implicaría complicar la forma en la que el módulo de visión hace las comparaciones entre posturas del robot y el paciente, por eso no se ha hecho para este primer prototipo. Este es el motivo por el que la arquitectura, actualmente, es ad-hoc para el robot NAO. Aunque sus componentes pueden ser reutilizados individualmente sin ningún problema en otra arquitectura RoboComp que implemente sus interfaces ICE.

5.3. Clasificador de expresiones faciales

En esta otra parte, se ha investigado la aplicabilidad del aprendizaje relacional en el reconocimiento de expresiones faciales y se ha desarrollado el prototipo de un sistema capaz de distinguir 9 de 10 expresiones diferentes. Se ha usado el algoritmo de minería de datos relacional Tilde, alcanzándose una precisión de 81 %. El principal criterio de diseño a lo largo de todo el desarrollo ha sido la idea de que Tilde tenía que ser capaz de aprender con la mínima información posible. Por eso se utilizaron solo 9 puntos característicos de la cara de un total de 121 y los predicados de la base de conocimiento se modelaron con la menor cantidad posible de conocimiento del dominio.

A pesar de todas estas restricciones, los resultados muestran que Tilde no solo ha conseguido clasificar bien la mayoría de las expresiones faciales, sino que los árboles de decisión que generan muestran relaciones complejas y muy difíciles de ver a simple vista.

La principal dificultad de este trabajo radica en que las variaciones entre las distintas expresiones pueden ser muy sutiles o muy grandes. Seleccionar los puntos adecuados y diseñar solo unos pocos predicados que aporten la información correcta para todos los tipos de expresiones requiere mucha evaluación empírica. Aunque este sistema sea solo una primera aproximación, los resultados son prometedores.

Todavía hay mucho margen de mejora para este sistema, ya que solo es una prueba de la viabilidad de las técnicas relacionales en este ámbito. Lo más inmediato para aumentar la precisión sería probar la discretización interna de ACE-Tilde y reajustar la normalización de distancias y ángulos en función de los resultados que se han observado. Además, ahora que queda claro que Tilde funciona para clasificar expresiones faciales, es razonable aprovechar más el conocimiento del dominio que se tiene. A continuación se propone una lista de ideas concretas sobre las que se puede empezar a trabajar:

- Discretizar automáticamente usando las funciones internas de ACE-Tilde, ya que utiliza métodos de máxima entropía que podrían ser muy adecuados para este problema de clasificación particular.
- Calcular los ángulos sobre la recta que pasa por los puntos de los ojos, en lugar de sobre el eje X. Así esta medida sería mucho más robusta ante ciertos giros de la cabeza.
- Añadir nuevos tipos de predicados.
- Experimentar con más personas.
- Encontrar el modo de que las mallas se adapten a caras tristes o directamente usar otras mallas no proporcionadas por el Face Tracker SDK de Microsoft.
- Encontrar el modo de obtener las coordenadas en 3D del Modelo de Malla, en lugar de su proyección en 2D.
- Extenderlo a un clasificador multietiqueta-multiclase.
- Añadir más expresiones faciales diferentes, similares a las que usan los otros trabajos del estado de la cuestión para poder comparar mejor los resultados de este trabajo con los suyos.

- Hacer el *matching* del árbol relacional que genera Tilde para poder clasificar las caras del vídeo en tiempo real.
- En la validación, se podrían tener en cuenta aspectos temporales (entre fotogramas) para mejorar la precisión. Por ejemplo, en un vídeo las expresiones nunca durarán solo un fotograma, por lo que éstas podrían considerarse errores de clasificación. Bastaría añadir un retardo de décimas de segundo a la hora de mostrar el resultado para detectar estos errores y omitirlos.
- Automatizar el proceso de aprendizaje y entrenamiento con Tilde para que el usuario indique en qué momento quiere aprender una cara nueva y el sistema sea capaz de reconocerla a partir de ahí.
- Integrarlo en el proyecto NAOTherapist y THERAPIST junto con un reconocimiento de posturas corporales para tener un módulo de atención del usuario en tiempo real.

5.4. NAOTherapist

El prototipo inicial está terminado y la prueba de concepto funciona. Sin embargo aún queda trabajo para poder hacer pruebas y evaluaciones de usuarios en un entorno real.

Una segunda fase del desarrollo podría incluir el diseño e implementación de más interfaces de usuario que permitan configurar y gestionar mejor la base de conocimiento. Actualmente añadir nuevos ejercicios es muy confuso dado el formato de la base de conocimiento y los tiempos de cada postura. Una interfaz de usuario que permita controlar el aprendizaje de ejercicios y poses para que se almacenen automáticamente dentro de la base de conocimiento es básico para que el sistema sea fácil de usar. Un gestor de ejercicios almacenados en la base de conocimiento también sería muy útil para el terapeuta, ya que podría ver qué objetivos terapéuticos tienen más representación en la base de conocimiento y cuales menos.

Otra posible ampliación es el registro y almacenamiento en el perfil del paciente de variables terapéuticas que faciliten el seguimiento de los pacientes por parte de los médicos y pueda dar información útil a los ingenieros para mejorar los componentes de la arquitectura. También sería necesaria algún tipo de interfaz de usuario para que los terapeutas y los médicos pudieran consultar y editar esta información de forma sencilla.

La interacción social de NAOTherapist aún es limitada, aunque ésta no formaba parte del prototipo inicial ni de este trabajo. La inclusión de un módulo de atención del paciente permitiría detectar sus expresiones faciales (por ejemplo con el sistema de aprendizaje relacional explicado en este trabajo) y sus posturas corporales (con otro sistema de lenguaje corporal que se estaba desarrollando paralelamente al de reconocimiento facial). Esta información serviría de contexto para un futuro componente conversacional que permitiese interactuar socialmente con el paciente de forma mucho más variada y coherente. El módulo de atención también permitiría al componente Ejecutivo añadir nuevos eventos exógenos para que PELEA planificase la acción de "llamar atención del paciente" (claim-attention), por ejemplo.

Con estas mejoras en la interacción social y en la usabilidad para el equipo médico, la principal tarea que se debería hacer es una evaluación con pacientes y equipo médico real. La evaluación de todo el sistema y de la generación de terapias por separado sería una información básica para determinar qué aspectos deberían mejorarse y en cuáles se ha acertado.

Fuera del marco de lo que podría considerarse un prototipo, en este proyecto también tiene cabida una avanzada función de aprendizaje de conceptos de la cual, actualmente, solo se tiene un esbozo. El componente de aprendizaje podría aprender nuevos conceptos que modificasen los dominios PDDL para incluirlos en la planificación. También podría conectarse al Ejecutivo para que éste le enviase información sobre el plan y qué acciones tienen más éxito según el estado del mundo actual (lo que en planificación se denomina reutilización de políticas).

Por supuesto, la reutilización de los componentes en el proyecto THERAPIST es otro de los objetivos fundamentales. Generalizar la arquitectura ad-hoc de NAOTherapist para que sea independiente del robot es el siguiente paso a seguir para que THERAPIST sea una realidad.

Bibliografía

[Ahmed et al. 2010] Ahmed, S., Gozbasi, O., Savelsbergh, M. W. P., Crocker, I., Fox, T., and Schreibmann, E. (2010). An automated intensity-modulated radiation therapy planning system. *INFORMS Journal on Computing*, 22(4):568–583. (Citada en pág. 22)

[Alcázar et al. 2010] Alcázar, V., Guzmán, C., Prior, D., Borrajo, D., Castillo, L., and Onaindia, E. (2010). Pelea: Planning, learning and execution architecture. In *28th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2010)*. (Citada en pág. 24)

[Bartlett et al. 2006] Bartlett, M. S., Littlewort, G. C., Frank, M. G., Lainscsek, C., Fasel, I. R., and Movellan, J. R. (2006). Automatic Recognition of Facial Actions in Spontaneous Expressions. *Journal of Multimedia*, 1(6):22–35. (Citada en pág. 26)

[Benton et al. 2012] Benton, J., Coles, A. J., and Coles, A. I. (2012). Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS-12)*. (Citada en pág. 23 y 83)

[Bettadapura 2012] Bettadapura, V. (2012). Face Expression Recognition and Analysis: The State of the Art. *Tech report*, pages 1–27. (Citada en pág. 10 y 25)

[Borggraefe et al. 2010] Borggraefe, I., Kiwull, L., Schaefer, J. S., Koerte, I., Blaschek, A., Meyer-Heim, A., and Heinen, F. (2010). Sustainability of motor performance after

robotic-assisted treadmill therapy in children: an open, non-randomized baseline-treatment study. *Eur J Phys Rehabil Med.* (Citada en pág. 8)

[Burgar et al. 2000] Burgar, C., Lum, P., Shor, P., and Machiel, V. d. L. (2000). Development of robots for rehabilitation therapy: the palo alto va/stanford experience. In *Journal of Rehabilitation Research and Development Vol. 37 No. 6*, pages 663–673. (Citada en pág. 13)

[Calderita et al. 2013] Calderita, L. V., Bustos García de Castro, P., Suárez Mejías, C., Fernández Rebollo, F., and Bandera Rubio, A. (2013). THERAPIST: Towards an Autonomous Socially Interactive Robot for Motor and Neurorehabilitation Therapies for Children. In *Proceedings of the ICTs for improving Patients Rehabilitation Research Techniques*, pages 374–377. IEEE. (Citada en pág. 3, 6 y 16)

[Castelli 2011] Castelli, E. (2011). Robotic movement therapy in cerebral palsy. *Dev Med Child Neurol.* (Citada en pág. 2)

[Choe et al. 2013] Choe, Y.-k., Jung, H.-T., Baird, J., and Grupen, R. A. (2013). Multidisciplinary stroke rehabilitation delivered by a humanoid robot: Interaction between speech and physical therapies. *Aphasiology*, 27(3):252–270. (Citada en pág. 15)

[Drużbicki et al. 2013] Drużbicki, M., Rusek, W., Snela, S., Dudek, J., Szczepanik, M., Zak, E., Durmala, J., Czernuszenko, A., Bonikowski, M., and Sobota, G. (2013). Functional effects of robotic-assisted locomotor treadmill thearapy in children with cerebral palsy. *J Rehabil Med.* (Citada en pág. 8)

[Dzeroski 2005] Dzeroski, S. (2005). Relational data mining. In Maimon, O. and Rokach, L., editors, *The Data Mining and Knowledge Discovery Handbook*, pages 869–898. Springer. (Citada en pág. 10)

[Eriksson et al. 2005] Eriksson, J., Matarić, M. J., and Winstein, C. J. (2005). Hands-off assistive robotics for post-stroke arm rehabilitation. In *In Proceedings of the International Conference on Rehabilitation Robotics.* (Citada en pág. 13 y 14)

[Fasola et al. 2010] Fasola, J. and Mataric, M. (2010). Robot exercise instructor: A socially assistive robot system to monitor and encourage physical exercise for the elderly. In *RO-MAN, 2010 IEEE*, pages 416–421. (Citada en pág. 15)

[Fdez-Olivares et al. 2011] Fdez-Olivares, J., Castillo, L. A., Cózar, J. A., and García-Pérez, Ó. (2011). Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence*, 27(1):103–122. (Citada en pág. 22)

[Feil-seifer et al. 2005] Feil-seifer, D. and Matarić, M. J. (2005). Defining socially assistive robotics. In *in Proc. IEEE International Conference on Rehabilitation Robotics (ICORR'05*, pages 465–468. (Citada en pág. 13)

[Fox et al. 2003] Fox, M. and Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.(JAIR)*, 20:61–124. (Citada en pág. 23 y 43)

[Fridin et al. 2011] Fridin, M., Bar-Haim, S., and Belokopytov, M. (2011). Robotics agent coacher for cp motor function (rac cp fun). In *Robotics for Neurology and Rehabilitation, IROS International Conference on Intelligent Robots and Systems.* (Citada en pág. 8 y 15)

[Fuentetaja et al. 2010] Fuentetaja, R., Borrajo, D., and López, C. L. (2010). A look-ahead B&B search for cost-based planning. In *Current Topics in Artificial Intelligence*, pages 201–211. Springer. (Citada en pág. 23, 43 y 67)

[Ghallab et al. 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice*. Elsevier. (Citada en pág. 7, 22 y 36)

[González-Ferrer et al. 2013] González-Ferrer, A., Ten Teije, A., Fdez-Olivares, J., and Milian, K. (2013). Automated generation of patient-tailored electronic care pathways by translating computer-interpretable guidelines into hierarchical task networks. *Artificial Intelligence in Medicine*, 57(2):91–109. (Citada en pág. 22)

[Graf et al. 2009] Graf, B., Reiser, U., Hägele, M., Mauz, K., and Klein, P. (2009). Robotic home assistant care-o-bot®3 - product vision and innovation platform. In *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, pages 139–144. (Citada en pág. 13)

[Hensey 2009] Hensey, O. (2009). Cerebral palsy: Challenges and opportunities. In *Technical report, State Claims Agency.* (Citada en pág. 2)

[Hoffmann 2003] Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341. (Citada en pág. 23 y 67)

[Jamshidnezhad et al. 2011] Jamshidnezhad, A. and jan Nordin, M. (2011). A Classifier Model based on the Features Quantitative Analysis for Facial Expression Recognition. In *Proceedings of the International Conference on Advanced Science, Engineering and Information Technology*, pages 391–394. (Citada en pág. 26)

[Kahn et al. 2001] Kahn, L. E., Averbuch, M., Rymer, W. Z., Reinkensmeyer, D. J., and D, P. (2001). Comparison of robot-assisted reaching to free reaching in promoting recovery from chronic stroke. In *In Integration of Assistive Technology in the Information Age, Proceedings 7th International Conference on Rehabilitation Robotics*, pages 39–44. IOS Press. (Citada en pág. 13)

[Krägeloh-Mann et al. 2009] Krägeloh-Mann, I. and Cans, C. (2009). Cerebral palsy update. *Brain Dev*, 31(7):537–44. (Citada en pág. 2)

[Limthongthang et al. 2013] Limthongthang, R., Bachoura, A., Songcharoen, P., and Osterman, A. L. (2013). Adult brachial plexus injury: evaluation and management. *The Orthopedic clinics of North America*, 44(4):591–603. (Citada en pág. 2)

[Manso et al. 2010] Manso, L., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and Calderita, L. (2010). Robocomp: A tool-based robotics framework. In Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., and von Stryk, O., editors, *Simulation,*

Modeling, and Programming for Autonomous Robots, volume 6472 of *Lecture Notes in Computer Science*, pages 251–262. Springer Berlin Heidelberg. (Citada en pág. 9 y 20)

[Matarić et al. 2007] Matarić, M. J., Eriksson, J., Feil-Seifer, D. J., and Winstein, C. J. (2007). Socially assistive robotics for post-stroke rehabilitation. *Journal of Neuro-Engineering and Rehabilitation*, 4(5). (Citada en pág. 8 y 14)

[McMurrough et al. 2012] McMurrough, C., Ferdous, S., Papangelis, A., Boisselle, A., and Makedon, F. (2012). A survey of assistive devices for cerebral palsy patients. In Makedon, F., editor, *PETRA*, page 17. ACM. (Citada en pág. 14)

[Meyer-Heim et al. 2013] Meyer-Heim, A. and van Hedel, H. J. A. (2013). Robot-assisted and computer-enhanced therapies for children with cerebral palsy: current state and clinical implementation. *Semin Pediatr Neurol*, 20(2):139–45. (Citada en pág. 3)

[Morignot et al. 2010] Morignot, P., Soury, M., Leroux, C., Vorobieva, H., and Hède, P. (2010). Generating scenarios for a mobile robot with an arm: Case study: Assistance for handicapped persons. In *Proceedings of 11th International Conference on Control Automation Robotics & Vision*, pages 976–981. IEEE. (Citada en pág. 22)

[Nalin et al. 2012] Nalin, M., Baroni, I., and Sanna, A. (2012). A Motivational Robot Companion for Children in Therapeutic Setting. In *IROS 2012*. (Citada en pág. 12)

[Peleg 2013] Peleg, M. (2013). Computer-interpretable clinical guidelines: A methodological review. *Journal of Biomedical Informatics*, 46(4):744–763. (Citada en pág. 22)

[Pulido et al. 2014] Pulido, J. C., González, J. C., González, A., García, J., Fernández, F., Bandera, A., Bustos, P., and Suárez, C. (2014). Goal-directed Generation of Exercise Sets for Upper-Limb Rehabilitation. In Barták, R., Fratini, S., McCluskey, L., and Vaquero, T., editors, *Proceedings of the 5th Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), ICAPS conference*, pages 38–45, Portsmouth (New Hampshire, USA). (Citada en pág. III, IV, 8 y 36)

[Ros et al. 2011] Ros, R., Nalin, M., Wood, R., Baxter, P., Looije, R., Demiris, Y., Belpaeme, T., Giusti, A., and Pozzi, C. (2011). Child-robot interaction in the wild: advice to the aspiring experimenter. In Bourlard, H., Huang, T. S., Vidal, E., Gatica-Perez, D., Morency, L.-P., and Sebe, N., editors, *ICMI*, pages 335–342. ACM. (Citada en pág. 8)

[Rosen et al. 2007] Rosen, J. and Perry, J. C. (2007). Upper limb powered exoskeleton. (Citada en pág. 13)

[Schimmelpfeng et al. 2012] Schimmelpfeng, K., Helber, S., and Kasper, S. (2012). Decision support for rehabilitation hospital scheduling. *OR spectrum*, 34(2):461–489. (Citada en pág. 22)

[Suárez Mejías et al. 2013] Suárez Mejías, C., Echevarría, C., Nuñez, P., Manso, L., Bustos, P., Leal, S., and Parra, C. (2013). Ursus: A robotic assistant for training of children with motor impairments. In Pons, J. L., Torricelli, D., and Pajaro, M., editors, *Converging Clinical and Engineering Research on Neurorehabilitation*, volume 1 of *Biosystems & Biorobotics*, pages 249–253. Springer Berlin Heidelberg. (Citada en pág. 16)

[Sumathi et al. 2012] Sumathi, C. P., Santhanam, T., and Mahadevi, M. (2012). Automatic Facial Expression Analysis A Survey. *International Journal of Computer Science & Engineering Survey*, 3(6):47–59. (Citada en pág. 25)

[Tapus et al. 2006] Tapus, A. and Matarić, M. J. (2006). Towards socially assistive robotics. (Citada en pág. 14)

[Tapus et al. 2007] Tapus, A., Matarić, M. J., and Scassellati, B. (2007). The grand challenges in socially assistive robotics. *IEEE Robotics and Automation Magazine*, 14(1):35–42. (Citada en pág. 12)

[Thai et al. 2011] Thai, L. H., Nguyen, N. D. T., and Hai, T. S. (2011). A Facial Expression Classification System Integrating Canny, Principal Component Analy-

sis and Artificial Neural Network. *International Journal of Machine Learning and Computing*, 1(4):388–393. (Citada en pág. 26)

[Turner-Stokes 2009] Turner-Stokes, L. (2009). Goal attainment scaling (GAS) in rehabilitation: a practical guide. *Clinical rehabilitation*, 23(4):362–70. (Citada en pág. 6 y 42)

[Yang et al. 2009] Yang, P., Liu, Q., and Metaxas, D. N. (2009). Boosting encoded dynamic features for facial expression recognition. *Pattern Recognition Letters*, 30(2):132–139. (Citada en pág. 26)

[Younes et al. 2004] Younes, H. and Littman, M. (2004). Ppddl1.0: An extension to pddl for expressing planning domains with probabilistic effects. (Citada en pág. 73)

[Zeng et al. 2009] Zeng, Z., Pantic, M., Roisman, G., and Huang, T. (2009). A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):39–58. (Citada en pág. 9)